TECHNICAL REFERENCE MANUAL

NVIDIA Tegra K1 Mobile Processor

Abstract

The Technical Reference Manual focuses on the logical organization and control of Tegra K1 mobile processors. It provides information for those modules that interface to external devices, or those that control fundamental chip operations. The modules detailed in this document provide an overview, any necessary programming guidelines, and a register listing for that module. Internal functional units such as video and graphics hardware acceleration are controlled by NVIDIA provided software and not documented here.

Revision History

Version	Date	Description
v01p	MAR 20, 2014	Public release to support Open Source Development. This document is a work in progress. There will be follow-up releases as new information is made available.
v02p	JUL 15, 2014	• Revised DSI and eDP maximum resolution to 3200x2000. • I2C Controller: Revised encoding for BC_TERMINATE field. • USB Complex: Revised maximum packet size supported on any endpoint to 1024 bytes. Clarified encoding of MEM_ALIGNMENT_MUX_EN fields.
v03p	OCT 15, 2014	• Activity Monitor/HDMI: New sections. • APB: Added “APB_MISC_GP_MIPI_PAD_CTRL_0”. Moved SATA_AUX registers to the SATA section. • AVP Cache Controller: Added “Initializing the AVP Cache Controller”. Added registers from offsets 0x18 through 0x84. • Clock and Reset Controller: Added “CLK_RST_CONTROLLER_PLLE_SS_CNTL_0” and “CLK_RST_CONTROLLER_PLLD2_SS_CFG_0”. Updated reset and default values of CLK_RST_CONTROLLER_PLLC3_MISC_0_0. • I2C Controller: Corrected encoding of I2C_I2C_BUS_CLEAR_CONFIG_0 register. • Interrupt Controller: Added “EVP Registers”. • MIPI-CSI (Camera Serial Interface): Updated the Pin Muxing for DSI and CSI Controllers figure. • Power Management Controller: Updated the description of the APBDEV_PMC_NO_IOPOWER_0 register. Added “Secure Boot Registers”. • SATA Controller: Added steps for System Power Management driver using software override to “ELPG Entry” and “ELPG Exit”. • SPI Controller: Added “Known Limitations” and “Special Guidelines for Slave Mode”. Clarified packet size in Unpacked Mode for Master and Slave modes. • Thermal Sensor: Added “Temperature Sensor Calibration Registers”.

Table of Contents

[THIS PAGE INTENTIONALLY LEFT BLANK]

1.0 INTRODUCTION

The NVIDIA® Tegra® K1 mobile processor is a complete applications and digital media system built around several powerful hardware elements:

- Graphics: NVIDIA® GeForce® Kepler Graphics Processing Unit (GPU). The GPU fully supports DX11, Shader Model 4, and OpenGL4.3 as well as OpenGL ES 3.0. It supports Unified shaders and is GPU compute capable with 192 CUDA cores. The GPU supports all the same features as discrete NVIDIA GPUs, including PhysX, CUDA, OpenCL, and DX compute. It is highly power optimized for best performance in mobile use cases.
- CPU Complex: Quad Cortex®-A15 Symmetric Multi-Processing ARM® Cores in a 4-PLUS-1™ configuration with a quad-core fast CPU complex and a fifth Battery Saver Core. The Cortex-A15 core features triple instruction issue and both out-of-order and speculative execution. It has full cache coherency support for the quad symmetric processors. All processors have 32 KB Instruction and 32 KB Data Level 1 caches; and there is a 2 MB shared Level 2 cache for the quad-core complex and a 512 KB Level 2 cache for the fifth core. The NVIDIA 4-PLUS-1 architecture uses the fifth Battery Saver Core, which operates exclusively with the main CPU complex, for very low-power, low-leakage operation at the light CPU loads common to multimedia and lightly loaded use situations.
- Memory Controller: 64-bit DRAM interface providing high bandwidth. LP-DDR3 and DDR3L DRAM types are supported.
- Audio/Video Decoder: the Audio-Video Processor (AVP) subsystem includes dedicated audio and video decode hardware acceleration, an ARM7 processor, and embedded RAM. This subsystem provides full motion playback of up to 1440P high-definition video and supports H.264 BP/MP/HP/MVC, VC-1, VP8, MPEG-2 and MPEG-4 video standards and multiple audio standards with dedicated hardware.
- Video Encoder: A high performance H.264 capable hardware video encoder. This processor supports H.264 BP/MP/HP/MVC and VP8 encoding.
- Imaging: A high-quality hardware accelerated still-image and video capture path, with dual next-generation ISP3s.
- Display: Dual display controllers with MIPI-DSI output, along with LVDS or eDP support for LCD panels and HDMI output for external display devices. Multiple line pixel storage allows more memory-efficient scaling operations and pixel fetching. Hardware display surface rotation is also provided for bandwidth reduction.

In addition to these major elements, Tegra K1 mobile processors have a broad range of peripheral interfaces to enable communication with wireless baseband, other communications peripherals, audio codecs, power management, and mass storage. When combined with baseband and PMIC chips, the Tegra K1 mobile processor provides the functionality needed to build a range of low-power devices. Dedicated high-performance mass storage controllers, with their own DMA engines, free the CPU Complex from routine data management tasks.

1.1 The Role of the Technical Reference Manual

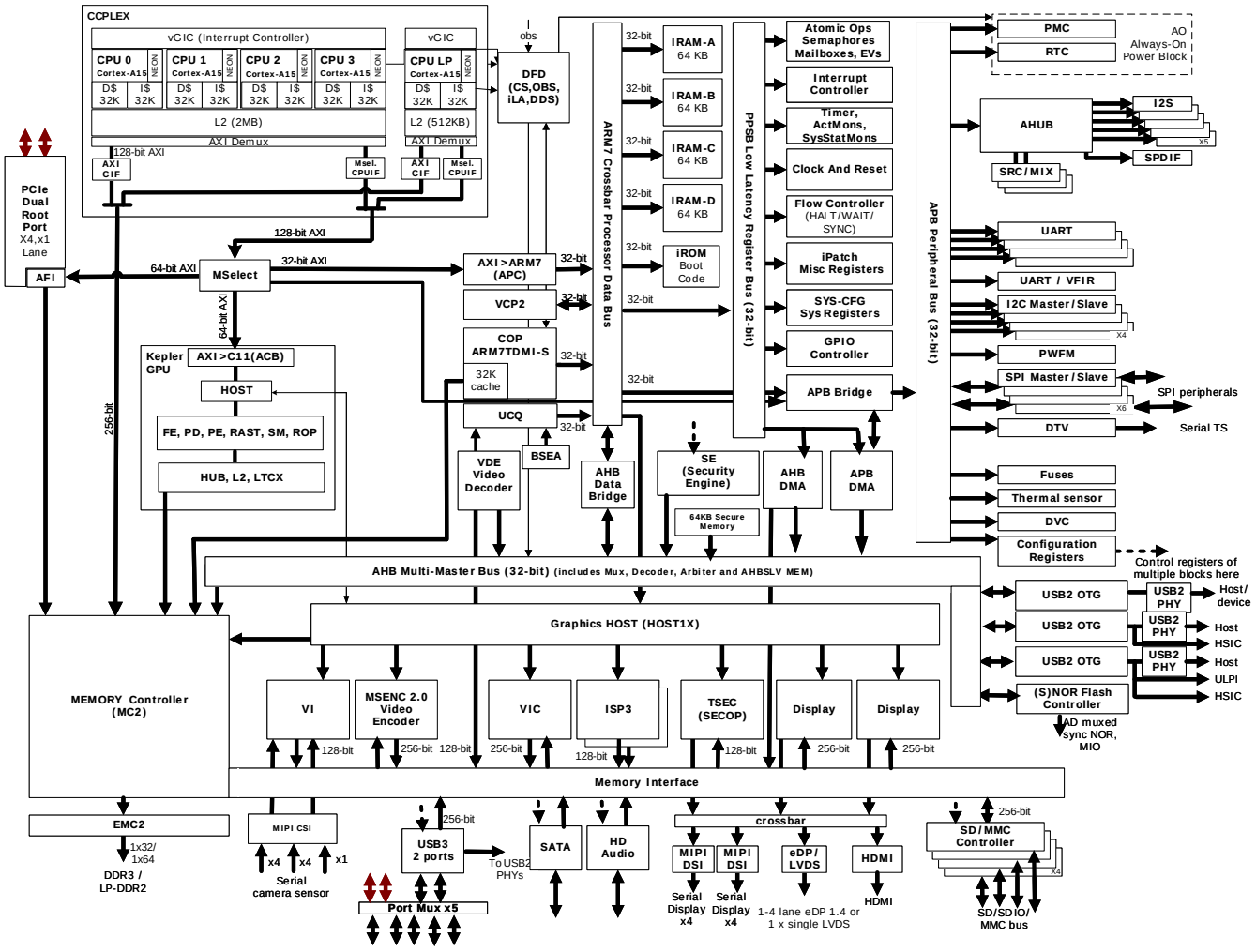
This document is intended to provide guidance to programmers writing code for Tegra K1 devices. It describes the register interfaces and hardware functionality, with the goal to aid anyone in understanding and potentially modifying the NVIDIA provided code.

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

1.2 Block Diagram

This diagram provides an overview of a Tegra K1 mobile processor.

Figure 1: Tegra K1 Processor Block Diagram



1.3 Memory Controller and Internal Bus Architecture

The Tegra K1 mobile processor has a highly optimized 64-bit memory controller, supporting low latency access for the CPU, optimized high bandwidth access for the graphics and video devices, and controlled latency for real time devices such as display.

There is a three-level hierarchy of memory clients:

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

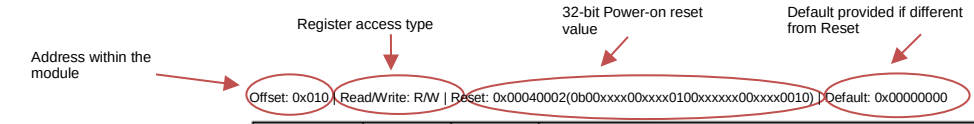
Special provisions are made for the CPU to bypass parts of the memory controller arbitration to help achieve a lower latency.

1.4 Reading Register Tables

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

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

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



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

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

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

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

1.5 Glossary

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

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

Term	Definition
IDE	Integrated Drive Electronics (or Integrated Device Electronics)
ISP	Image Signal Processor, a hardware engine that is part of the camera processing pipeline.
KBC	Keyboard Controller module allows the Tegra processor to be connected to keyboard matrices of sizes up to 11x8.
LVDS	Low Voltage Differential Signaling
MC	Memory Controller module handles requests from internal clients and arbitrates among them to allocate memory bandwidth.
MCCIF or MC-CIF	Memory Controller Client InterFace, the standard interface block between the memory controller sub-system fabric and the client device. Note that some modules may have multiple client interfaces.
MIPI	The Mobile Industry Processor Interface and industry alliance promoting a number of standard interfaces for mobile devices.
MPCORE	Multi-processor CPU core, a generic term for a CPU capable of operating as part of an SMP group.
MPE	An older name for the Video Encoder in the Tegra processor capable of encoding raw video stream into MPEG. Now referred to as MSENC.
MSENC	Multi-Standard video Encoder engine.
NAND	A type of flash memory supporting high densities, and commonly used for non-volatile mass storage in portable devices. Accessed in sequential blocks to be generally treated as a file system.
NOR	A type of flash memory, with a direct bus interface that allows random access so code can be executed in place. Generally more costly and less dense than NAND flash memory.
OGL	Open Graphics Library, also known as OpenGL. An API supported on Tegra devices and accelerated in hardware by dedicated 3D and 2D engines.
PCIe	Peripheral Component Interconnect Express, a high-speed interface for external devices connected to the Tegra SOC.
PMC	Power Management Controller module controls the various power management features in the system.
PPSB	PortalPlayer System Bus, a proprietary register bus used for some blocks. Similar to APB. PortalPlayer is a company that was acquired by NVIDIA, and from where parts of Tegra are derived including this bus.
PWFM	Pulse Width Frequency Modulation module generates programmed pulse widths typically used to control backlight in display panels.
PVT	Process, Voltage, & Temperature
RISC	Reduced Instruction Set Computer, the CPU architecture used by ARM CPUs.
SATA	Serial Advanced Technology Attachment (ATA)
SDMMC	SD and MMC controller. An I/O controller supporting
SLINK	Serial Link, a legacy and now obsolete name for the SPI controller
SMMU	System Memory Management Unit, a block within the memory controller used to map from a virtual address space to physical addresses for device DMA.
SMP	Symmetric Multi-Processing
SOC	System On a Chip, an integrated circuit containing a CPU, memory controller and the peripheral devices needed for a computing system.
SOR	Serial Output Resource. SOR is GPU IP for driving HDMI/DP/LVDS. It converts the output of the display to a more modern high-speed serial protocol. DSI is not included since it's not GPU IP based
S/PDIF	Sony/Philips Digital Interconnect Format
SPI	Serial Peripheral Interface Bus, a synchronous serial data link, that operates in full duplex mode.
TSEC	Tegra Security co-processor, an embedded security processor used mainly to manage the HDCP encryption and keys on the HDMI link.
TZ	Trust Zone, a secure operating environment of the ARM CPU and the related secure parts of the SOC backbone and devices

Term	Definition
TZRAM	Trust Zone secured RAM on the SOC.
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, a Tegra hardware acceleration block dedicated to decoding compressed video in various formats.
VI2	Video Input 2 block, the acronym used to describe the Tegra K1 block used for camera and related input functions.
VIC	Video Image Compositer, a Tegra K1 block that implements video post-processing functions needed by a video playback application to produce the final image for the player window.

[THIS PAGE INTENTIONALLY LEFT BLANK]

2.0 ADDRESS MAP

2.1 System Address Map

Table 1: System Address Map

Description	Address Start	Address End	Offset Start	Offset End	Default Length
Remap	0100:0000	3fff:ffff			1008 MB
PCIE	0100:0000	3fff:ffff	0000:0000	3eff:ffff	1008 MB
PCIE_A1	0100:0000	01ff:ffff			16 MB
PCIE_A2	0200:0000	0fff:ffff			224 MB
PCIE_A3	1000:0000	3fff:ffff			768 MB
Data Memory	**4000:0000**	**40ff:ffff**			**16 MB**
iRAM-A	4000:0000	4000:ffff			64 KB
iRAM-B	4001:0000	4001:ffff			64 KB
iRAM-C	4002:0000	4002:ffff			64 KB
iRAM-D	4003:0000	4003:ffff			64 KB
NOR Flash	**4800:0000**	**4fff:ffff**	0000:0000	07ff:ffff	**128 MB**
NOR Flash_A1	4800:0000	48ff:ffff			16 MB
NOR Flash_A2	4900:0000	4aff:ffff			32 MB
NOR Flash_A3	4b00:0000	4fff:ffff			80 MB
Graphics Host Registers	**5000:0000**	**5003:3fff**			**208 KB**
Host1x	5000:0000	**5003:3fff**	0000:0000	0003:3fff	208 KB
ARM registers (@ PERIPHBASE)	**5004:0000**	**5005:ffff**			**128 KB**
ARM PERIPHBASE	5004:0000	5005:ffff	0000:0000	0001:ffff	128 KB
ARM Interrupt Distributor	5004:1000	5004:1fff	0000:1000	0000:1fff	4 KB
AVP CACHE	5004:0000	5004:1fff	0000:0000	0000:1fff	8 KB
Interrupt Controller Physical CPU interface	5004:2000	5004:3fff	0000:2000	0000:3fff	8 KB
Interrupt Controller Virtual CPU interface (Hypervisor view for requesting CPU)	5004:4000	5004:4fff	0000:4000	0000:4fff	4 KB
Interrupt Controller Virtual CPU interface (Hypervisor view for all CPUs)	5004:5000	5004:5fff	0000:5000	0000:5fff	4 KB
Interrupt Controller Virtual CPU interface (Virtual Machine view)	5004:6000	5004:7fff	0000:6000	0000:7fff	8 KB
MSelect	**5006:0000**	**5006:0fff**			**4 KB**
MSelect Register Space	5006:0000	5006:0fff			4 KB
Graphics Host	**5400:0000**	**54ff:ffff**			**16 MB**
VI	5408:0000	540b:ffff	0008:0000	000b:ffff	256 KB
CSI	5408:0000	540b:ffff	0008:0000	000b:ffff	256 KB
ISP	5460:0000	5463:ffff	0060:0000	0063:ffff	256 KB
ISPB	5468:0000	546b:ffff	0068:0000	006b:ffff	256 KB
Display A	5420:0000	5423:ffff	0020:0000	0023:ffff	256 KB
Display B	5424:0000	5427:ffff	0024:0000	0027:ffff	256 KB
HDMI	5428:0000	542b:ffff	0028:0000	002b:ffff	256 KB
DSI	5430:0000	5433:ffff	0030:0000	0033:ffff	256 KB
VIC	5434:0000	5437:ffff	0034:0000	0037:ffff	256 KB

Description	Address Start	Address End	Offset Start	Offset End	Default Length
DSIB	5440:0000	5443:ffff	0040:0000	0043:ffff	256 KB
MSENC	544c:0000	544f:ffff	004c:0000	004f:ffff	256 KB
TSEC	5450:0000	5453:ffff	0050:0000	0053:ffff	256 KB
SOR	5454:0000	5457:ffff	0054:0000	0057:ffff	256 KB
DPAUX	545c:0000	545f:ffff	005c:0000	005f:ffff	256 KB
GPU	**5700:0000**	**5fff:ffff**	0000:0000	08ff:ffff	**144 MB**
GPU_GART	**5700:0000**	**5fff:ffff**			**144 MB**
PPSB	**6000:0000**	**60ff:ffff**			**16 MB**
uP-TAG	6000:0000	6000:0fff			4 KB
Resource Semaphore	6000:1000	6000:1fff	0000:0000	0000:0fff	4 KB
Arbitration Semaphore	6000:2000	6000:2fff	0000:1000	0000:1fff	4 KB
ARB-PRI	6000:3000	6000:3fff	0000:3000	0000:3fff	4 KB
Primary ICTLR	6000:4000	6000:403f	0000:4000	0000:403f	64 B
Primary ICTLR ARB-GNT	6000:4040	6000:40ff	0000:4040	0000:40ff	192 B
Secondary ICTLR	6000:4100	6000:41ff	0000:4100	0000:41ff	256 B
Tertiary ICTLR	6000:4200	6000:42ff	0000:4200	0000:42ff	256 B
Quad ICTLR	6000:4300	6000:43ff	0000:4300	0000:43ff	256 B
Penta ICTLR	6000:4400	6000:44ff	0000:4400	0000:44ff	256 B
HIER GROUP1 ICTLR	6000:4800	6000:48ff	0000:4800	0000:48ff	256 B
TMR	**6000:5000**	**6000:53ff**	**0000:5000**	**0000:53ff**	**1 KB**
TMR1			0000:0000	0000:0007	8 B
TMR2			0000:0008	0000:000f	8 B
TMRUS			0000:0010	0000:004f	64 B
TMR3			0000:0050	0000:0057	8 B
TMR4			0000:0058	0000:005f	8 B
TMR5			0000:0060	0000:0067	8 B
TMR6			0000:0068	0000:006f	8 B
TMR7			0000:0070	0000:0077	8 B
TMR8			0000:0078	0000:007f	8 B
TMR9			0000:0080	0000:0087	8 B
TMR0			0000:0088	0000:008f	8 B
WDT0			0000:0100	0000:011f	32 B
WDT1			0000:0120	0000:013f	32 B
WDT2			0000:0140	0000:015f	32 B
WDT3			0000:0160	0000:017f	32 B
WDT4			0000:0180	0000:019f	32 B
TMR_SHARED			0000:01a0	0000:01bf	32 B
Clock and Reset	6000:6000	6000:6fff	0000:6000	0000:6fff	4 KB
Flow Controller	6000:7000	6000:7fff	0000:7000	0000:7fff	4 KB
AHB-DMA	6000:8000	6000:9fff	0000:8000	0000:9fff	8 KB
AHB-DMA CH0	6000:9000	6000:901f	0000:9000	0000:901f	32 B
AHB-DMA CH1	6000:9020	6000:903f	0000:9020	0000:903f	32 B
AHB-DMA CH2	6000:9040	6000:905f	0000:9040	0000:905f	32 B
AHB-DMA CH3	6000:9060	6000:907f	0000:9060	0000:907f	32 B
APB-DMA	6002:0000	6002:3fff	0002:0000	0002:3fff	16 KB
APB-DMA CH0	6002:1000	6002:103f	0002:1000	0002:103f	64 B

Description	Address Start	Address End	Offset Start	Offset End	Default Length
APB-DMA CH1	6002:1040	6002:107f	0002:1040	0002:107f	64 B
APB-DMA CH2	6002:1080	6002:10bf	0002:1080	0002:10bf	64 B
APB-DMA CH3	6002:10c0	6002:10ff	0002:10c0	0002:10ff	64 B
APB-DMA CH4	6002:1100	6002:113f	0002:1100	0002:113f	64 B
APB-DMA CH5	6002:1140	6002:117f	0002:1140	0002:117f	64 B
APB-DMA CH6	6002:1180	6002:11bf	0002:1180	0002:11bf	64 B
APB-DMA CH7	6002:11c0	6002:11ff	0002:11c0	0002:11ff	64 B
APB-DMA CH8	6002:1200	6002:123f	0002:1200	0002:123f	64 B
APB-DMA CH9	6002:1240	6002:127f	0002:1240	0002:127f	64 B
APB-DMA CH10	6002:1280	6002:12bf	0002:1280	0002:12bf	64 B
APB-DMA CH11	6002:12c0	6002:12ff	0002:12c0	0002:12ff	64 B
APB-DMA CH12	6002:1300	6002:133f	0002:1300	0002:133f	64 B
APB-DMA CH13	6002:1340	6002:137f	0002:1340	0002:137f	64 B
APB-DMA CH14	6002:1380	6002:13bf	0002:1380	0002:13bf	64 B
APB-DMA CH15	6002:13c0	6002:13ff	0002:13c0	0002:13ff	64 B
APB-DMA CH16	6002:1400	6002:143f	0002:1400	0002:143f	64 B
APB-DMA CH17	6002:1440	6002:147f	0002:1440	0002:147f	64 B
APB-DMA CH18	6002:1480	6002:14bf	0002:1480	0002:14bf	64 B
APB-DMA CH19	6002:14c0	6002:14ff	0002:14c0	0002:14ff	64 B
APB-DMA CH20	6002:1500	6002:153f	0002:1500	0002:153f	64 B
APB-DMA CH21	6002:1540	6002:157f	0002:1540	0002:157f	64 B
APB-DMA CH22	6002:1580	6002:15bf	0002:1580	0002:15bf	64 B
APB-DMA CH23	6002:15c0	6002:15ff	0002:15c0	0002:15ff	64 B
APB-DMA CH24	6002:1600	6002:163f	0002:1600	0002:163f	64 B
APB-DMA CH25	6002:1640	6002:167f	0002:1640	0002:167f	64 B
APB-DMA CH26	6002:1680	6002:16bf	0002:1680	0002:16bf	64 B
APB-DMA CH27	6002:16c0	6002:16ff	0002:16c0	0002:16ff	64 B
APB-DMA CH28	6002:1700	6002:173f	0002:1700	0002:173f	64 B
APB-DMA CH29	6002:1740	6002:177f	0002:1740	0002:177f	64 B
APB-DMA CH30	6002:1780	6002:17bf	0002:1780	0002:17bf	64 B
APB-DMA CH31	6002:17c0	6002:17ff	0002:17c0	0002:17ff	64 B
System Registers	6000:c000	6000:c2ff	0000:c000	0000:c2ff	768 B
AHB Arbitration + Gizmo Controller			0000:0000	0000:014f	336 B
AHB/APB Debug Bus			0000:0150	0000:01ff	176 B
Secure Boot			0000:0200	0000:02ff	256 B
STAT-MON	6000:c400	6000:c7ff	0000:c400	0000:c7ff	1 KB
Activity Monitor	6000:c800	6000:cbff	0000:c800	0000:cbff	1 KB
GPIO-1	6000:d000	6000:d0ff	0000:d000	0000:d0ff	256 B
GPIO-2	6000:d100	6000:d1ff	0000:d100	0000:d1ff	256 B
GPIO-3	6000:d200	6000:d2ff	0000:d200	0000:d2ff	256 B
GPIO-4	6000:d300	6000:d3ff	0000:d300	0000:d3ff	256 B
GPIO-5	6000:d400	6000:d4ff	0000:d400	0000:d4ff	256 B
GPIO-6	6000:d500	6000:d5ff	0000:d500	0000:d5ff	256 B
GPIO-7	6000:d600	6000:d6ff	0000:d600	0000:d6ff	256 B
GPIO-8	6000:d700	6000:d7ff	0000:d700	0000:d7ff	256 B
VCP	6000:e000	6000:efff	0000:e000	0000:efff	4 KB

Description	Address Start	Address End	Offset Start	Offset End	Default Length
Exception vectors	6000:f000	6000:ffff	0000:f000	0000:ffff	4 KB
AVPUCQ	6001:0000	6001:00ff	0001:0000	0001:00ff	256 B
BSEA	6001:1000	6001:1fff	0001:1000	0001:1fff	4 KB
IPATCH	6001:dc00	6001:dfff	0001:dc00	0001:dfff	1 KB
VDE	6003:0000	6003:3fff	0003:0000	0003:3fff	16 KB
SXE	6003:0000	6003:0fff	0003:0000	0003:0fff	4 KB
BSEV	6003:1000	6003:1fff	0003:1000	0003:1fff	4 KB
MBE	6003:2000	6003:20ff	0003:2000	0003:20ff	256 B
PPE	6003:2200	6003:22ff	0003:2200	0003:22ff	256 B
MCE	6003:2400	6003:24ff	0003:2400	0003:24ff	256 B
TFE	6003:2600	6003:26ff	0003:2600	0003:26ff	256 B
PPB	6003:2800	6003:28ff	0003:2800	0003:28ff	256 B
VDMA	6003:2a00	6003:2aff	0003:2a00	0003:2aff	256 B
UCQ	6003:2c00	6003:2cff	0003:2c00	0003:2cff	256 B
FRAMEID	6003:3800	6003:3bff	0003:3800	0003:3bff	1 KB
APB	**7000:0000**	**70ff:ffff**			**16 MB**
MISC	7000:0000	7000:3fff	0000:0000	0000:3fff	16 KB
PP			0000:0000	0000:03ff	1 KB
SC1X_PADS			0000:0400	0000:07ff	1 KB
GP			0000:0800	0000:0bff	1 KB
SECURE_REGS			0000:0c00	0000:0cff	256 B
OBS			0000:0d00	0000:0dff	256 B
USB_AUX			0000:1000	0000:10ff	256 B
SATA_AUX			0000:1100	0000:11ff	256 B
PINMUX_AUX			0000:3000	0000:3fff	4 KB
UART-A	7000:6000	7000:603f	0000:6000	0000:603f	64 B
UART-B	7000:6040	7000:607f	0000:6040	0000:607f	64 B
VFIR	7000:6100	7000:61ff	0000:6100	0000:61ff	256 B
UART-C	7000:6200	7000:62ff	0000:6200	0000:62ff	256 B
UART-D	7000:6300	7000:63ff	0000:6300	0000:63ff	256 B
HDMI_IOBIST	7000:6500	7000:65ff	0000:6500	0000:65ff	256 B
MIPI_IOBIST	7000:6600	7000:66ff	0000:6600	0000:66ff	256 B
LPDDR2_IOBIST	7000:6700	7000:67ff	0000:6700	0000:67ff	256 B
PCIE_X2_0_IOBIST	7000:6800	7000:68ff	0000:6800	0000:68ff	256 B
PCIE_X2_1_IOBIST	7000:6900	7000:69ff	0000:6900	0000:69ff	256 B
PCIE_X4_IOBIST	7000:6a00	7000:6aff	0000:6a00	0000:6aff	256 B
SATA_IOBIST	7000:6c00	7000:6dff	0000:6c00	0000:6dff	512 B
XIO Interface	7000:8a00	7000:8bff	0000:8a00	0000:8bff	512 B
Sync NOR	7000:9000	7000:9fff	0000:9000	0000:9fff	4 KB
PWM Controller	7000:a000	7000:a0ff	0000:a000	0000:a0ff	256 B
MIPIHSI Baseband	7000:b000	7000:bfff	0000:b000	0000:bfff	4 KB
I2C	7000:c000	7000:c0ff	0000:c000	0000:c0ff	256 B
TWC	7000:c100	7000:c1ff	0000:c100	0000:c1ff	256 B
DTV	7000:c300	7000:c3ff	0000:c300	0000:c3ff	256 B
I2C2	7000:c400	7000:c4ff	0000:c400	0000:c4ff	256 B
I2C3	7000:c500	7000:c5ff	0000:c500	0000:c5ff	256 B

Description	Address Start	Address End	Offset Start	Offset End	Default Length
OWR	7000:c600	7000:c6ff	0000:c600	0000:c6ff	256 B
I2C4	7000:c700	7000:c7ff	0000:c700	0000:c7ff	256 B
I2C5	7000:d000	7000:d0ff	0000:d000	0000:d0ff	256 B
I2C6	7000:d100	7000:d1ff	0000:d100	0000:d1ff	256 B
SPI 2B-1	7000:d400	7000:d5ff	0000:d400	0000:d5ff	512 B
SPI 2B-2	7000:d600	7000:d7ff	0000:d600	0000:d7ff	512 B
SPI 2B-3	7000:d800	7000:d9ff	0000:d800	0000:d9ff	512 B
SPI 2B-4	7000:da00	7000:dbff	0000:da00	0000:dbff	512 B
SPI 2B-5	7000:dc00	7000:ddff	0000:dc00	0000:ddff	512 B
SPI 2B-6	7000:de00	7000:dfff	0000:de00	0000:dfff	512 B
RTC	7000:e000	7000:e0ff	0000:e000	0000:e0ff	256 B
KBC	7000:e200	7000:e2ff	0000:e200	0000:e2ff	256 B
PMC	7000:e400	7000:ebff	0000:e400	0000:ebff	2 KB
FUSE	7000:f800	7000:fbff	0000:f800	0000:fbff	1 KB
KFUSE	7000:fc00	7000:ffff	0000:fc00	0000:ffff	1 KB
LA	7001:0000	7001:1fff	0001:0000	0001:1fff	8 KB
SE	7001:2000	7001:3fff	0001:2000	0001:3fff	8 KB
TSENSOR	7001:4000	7001:4fff	0001:4000	0001:4fff	4 KB
CEC	7001:5000	7001:5fff	0001:5000	0001:5fff	4 KB
ATOMICS	7001:6000	7001:7fff	0001:6000	0001:7fff	8 KB
MC	7001:9000	7001:9fff	0001:9000	0001:9fff	4 KB
EMC	7001:b000	7001:bfff	0001:b000	0001:bfff	4 KB
SATA	7002:0000	7002:ffff	0002:0000	0002:ffff	64 KB
HDA	7003:0000	7003:ffff	0003:0000	0003:ffff	64 KB
MIOBFM	7020:0000	7020:ffff	0020:0000	0020:ffff	64 KB
AUDIO_CLUSTER	7030:0000	7030:ffff	0030:0000	0030:ffff	64 KB
APBIF			0000:0000	0000:01ff	512 B
APBIF2			0000:0200	0000:07ff	1536 B
AUDIO			0000:0800	0000:0fff	2 KB
I2S0			0000:1000	0000:10ff	256 B
I2S1			0000:1100	0000:11ff	256 B
I2S2			0000:1200	0000:12ff	256 B
I2S3			0000:1300	0000:13ff	256 B
I2S4			0000:1400	0000:14ff	256 B
DAM0			0000:2000	0000:21ff	512 B
DAM1			0000:2200	0000:23ff	512 B
DAM2			0000:2400	0000:25ff	512 B
AMX0			0000:3000	0000:30ff	256 B
AMX1			0000:3100	0000:31ff	256 B
ADX0			0000:3800	0000:38ff	256 B
ADX1			0000:3900	0000:39ff	256 B
SPDIF			0000:6000	0000:60ff	256 B
AFC0			0000:7000	0000:70ff	256 B
AFC1			0000:7100	0000:71ff	256 B
AFC2			0000:7200	0000:72ff	256 B
AFC3			0000:7300	0000:73ff	256 B

Description	Address Start	Address End	Offset Start	Offset End	Default Length
AFC4			0000:7400	0000:74ff	256 B
AFC5			0000:7500	0000:75ff	256 B
XUSB_PADCTL	7009:f000	7009:ffff	0009:f000	0009:ffff	4 KB
XUSB_HOST	7009:0000	7009:9fff	0009:0000	0009:9fff	40 KB
XUSB_DEV	700d:0000	700d:9fff	000d:0000	000d:9fff	40 KB
DDS	700a:0000	700a:11ff	000a:0000	000a:11ff	4608 B
SDMMC-1	700b:0000	700b:01ff	000b:0000	000b:01ff	512 B
SDMMC-1B	700b:1000	700b:11ff	000b:1000	000b:11ff	512 B
SDMMC-2	700b:0200	700b:03ff	000b:0200	000b:03ff	512 B
SDMMC-2B	700b:2200	700b:23ff	000b:2200	000b:23ff	512 B
SDMMC-3	700b:0400	700b:05ff	000b:0400	000b:05ff	512 B
SDMMC-3B	700b:3400	700b:35ff	000b:3400	000b:35ff	512 B
SDMMC-4	700b:0600	700b:07ff	000b:0600	000b:07ff	512 B
SDMMC-4B	700b:4600	700b:47ff	000b:4600	000b:47ff	512 B
SPEEDO	700c:0000	700c:7fff	000c:0000	000c:7fff	32 KB
SPEEDO_0			0000:0000	0000:00ff	256 B
SPEEDO_1			0000:0100	0000:01ff	256 B
SPEEDO_PMON	700c:8000	700c:ffff	000c:8000	000c:ffff	32 KB
SPEEDO_PMON_0			0000:0000	0000:01ff	512 B
SPEEDO_PMON_1			0000:0200	0000:03ff	512 B
SYSCTR0	700f:0000	700f:ffff	000f:0000	000f:ffff	64 KB
SYSCTR1	7010:0000	7010:ffff	0010:0000	0010:ffff	64 KB
DP2	700e:0000	700e:00ff	000e:0000	000e:00ff	256 B
APB2JTAG	700e:1000	700e:11ff	000e:1000	000e:11ff	512 B
SOC_THERM	700e:2000	700e:2fff	000e:2000	000e:2fff	4 KB
MIPI_CAL	700e:3000	700e:30ff	000e:3000	000e:30ff	256 B
DVFS	7011:0000	7011:03ff	0011:0000	0011:03ff	1 KB
CLUSTER_CLOCK	7004:0000	7007:ffff	0004:0000	0007:ffff	256 KB
CSITE	7080:0000	709f:ffff	0080:0000	009f:ffff	2 MB
AHB_A2	**7c00:0000**	**7dff:ffff**			**32 MB**
PPCS (AHB to MC flush)	7c00:0000	7c00:ffff	0000:0000	0000:ffff	64 KB
TZRAM	7c01:0000	7c01:ffff	0001:0000	0001:ffff	64 KB
USB	7d00:0000	7d00:17ff	0100:0000	0100:17ff	6 KB
USB2	7d00:4000	7d00:57ff	0100:4000	0100:57ff	6 KB
USB3	7d00:8000	7d00:97ff	0100:8000	0100:97ff	6 KB
External Memory	**8000:0000**	**27fff:ffff**			**8 GB**
EMEM	8000:0000	27fff:ffff	0000:0000	1ffff:ffff	8 GB
IROM (boot code)	**0010:0000**	**0010:ffff**			**64 KB**
IROML	0010:0000	0010:ffff			**64 KB**
Lo-VEC	**0000:0000**	**0000:ffff**			**64 KB**
IROM_LOVEC	**0000:0000**	**00ff:ffff**			**16 MB**

2.2 Available DRAM Address Ranges

Table 2 describes the various address regions that are available as DRAM in Tegra® K1 devices.

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

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

Table 2: DRAM Address Ranges

Aperture		Range	Size (MB)	H/W Config Option	Physical DRAM Range				IOVA Range Available for:	
					2 GB	3GB	4 GB	>4 GB	COP/ AVP?	AHB Clients?
DRAM		8000_0000 - FFFF_FFFF	2048	Always DRAM	DRAM	DRAM	DRAM	DRAM	Yes	Yes
DRAM2		1_0000_0000 - 2_7FFF_FFFF	6144	Always DRAM	N/A	N/A	N/A	DRAM	No	No
AHB_A2_rsvd		7E00_0000 - 7FFF_FFFF	32	Selectable	DRAM+	DRAM+	DRAM	DRAM+	Yes	Yes
AHB_A2		7C00_0000 - 7DFF_FFFF	32	Always MMIO	MMIO	MMIO	MMIO	MMIO	No	No
AHB_A1_rsvd		7900_0000 - 7BFF_FFFF	48	Selectable	DRAM+	DRAM+	DRAM	DRAM+	Yes	Yes
AHB_A1		7800_0000 - 78FF_FFFF	16	Selectable	MMIO	MMIO	MMIO	MMIO	No	No
APB_rsvd		7100_0000 - 77FF_FFFF	112	Selectable	DRAM+	DRAM+	DRAM	DRAM+	Yes	Yes
APB		7000_0000 - 70FF_FFFF	16	Always MMIO	MMIO	MMIO	MMIO	MMIO	No	No
ExtIO_rsvd		6900_0000 - 6FFF_FFFF	112	Selectable	DRAM+	DRAM+	DRAM	DRAM+	Yes	Yes
ExtIO		6800_0000 - 68FF_FFFF	16	Always MMIO	MMIO	MMIO	MMIO	MMIO	No	No
PPSB_rsvd		6100_0000 - 67FF_FFFF	112	Selectable	DRAM+	DRAM+	DRAM	DRAM+	Yes	Yes
PPSB		6000_0000 - 60FF_FFFF	16	Always MMIO	MMIO	MMIO	MMIO	MMIO	No	No
GART/GPU_GART		5700_0000 - 5FFF_FFFF	144	Always MMIO	MMIO	MMIO	MMIO	MMIO	Yes	Yes
Graphics Host rsvd (HOST1X_DR_RSVD)		5500_0000 - 56FF_FFFF	32	Selectable	DRAM+	DRAM+	DRAM	DRAM+	Yes	Yes
Graphics Host (HOST1X_DR)		5400_0000 - 54FF_FFFF	16	Always MMIO	MMIO	MMIO	MMIO	MMIO	No	No
Verif Aper + Rsvd		5100_0000 - 53FF_FFFF	48	Selectable	DRAM+	DRAM+	DRAM	DRAM+	Yes	Yes
Host1x+PERIPH[1]		5000_0000 - 50FF_FFFF	16	Always MMIO	MMIO	MMIO	MMIO	MMIO		
Host1X+ PERIPH	Host1x	5000_0000 – 5002_7FFF	160KB	Always MMIO	MMIO	MMIO	MMIO	MMIO	No	No
	PERIPHBASE	5004_0000 - 5005_FFFF	128KB	Always MMIO	MMIO	MMIO	MMIO	MMIO	No	Yes
	MSELECT	5006_0000 - 5006_0FFF	4KB	Always MMIO	MMIO	MMIO	MMIO	MMIO	No	Yes
NOR_A3		4B00_0000 - 4FFF_FFFF	80	Selectable	DRAM/ MMIO	DRAM/ MMIO	DRAM/ MMIO	DRAM/ MMIO	Yes if no NOR.	Yes if no NOR.
NOR_A2		4900_0000 - 4AFF_FFFF	32	Selectable	DRAM/ MMIO	DRAM/ MMIO	DRAM/ MMIO	DRAM/ MMIO	Yes if no NOR.	Yes if no NOR.
NOR_A1		4800_0000 - 48FF_FFFF	16	Selectable	DRAM/ MMIO	DRAM/ MMIO	DRAM/ MMIO	DRAM/ MMIO	Yes if no NOR.	Yes if no NOR.
IRAM_rsvd		4100_0000 - 47FF_FFFF	112	Selectable	DRAM+	DRAM+	DRAM	DRAM+	Yes	Yes

[1] The Host1X+PERIPH is split into multiple apertures.

Aperture	Range	Size (MB)	H/W Config Option	Physical DRAM Range				IOVA Range Available for:	
				2 GB	3GB	4 GB	>4 GB	COP/ AVP?	AHB Clients?
IRAM	4000_0000 - 40FF_FFFF	16	Always MMIO	MMIO	MMIO	MMIO	MMIO	No	No
PCIE_A3	1000_0000 - 3FFF_FFFF	768	Selectable	DRAM/ MMIO	DRAM/ MMIO	DRAM/ MMIO	DRAM/ MMIO	Yes	Yes
PCIE_A2	0200_0000 - 0FFF_FFFF	224	Selectable	DRAM/ MMIO	DRAM/ MMIO	DRAM/ MMIO	DRAM/ MMIO	Yes	Yes
PCIE_A1	0100_0000 - 01FF_FFFF	16	Selectable	DRAM/ MMIO	DRAM/ MMIO	DRAM/ MMIO	DRAM/ MMIO	Yes	Yes
IROM_LOVEC	0000_0000 - 00FF_FFFF	16	Always MMIO	MMIO	MMIO	MMIO	MMIO	No	No

Notes:

- The IOVA Range column is available for AVP or AHB clients that have SMMU translation enabled or PA address for AHB clients that have SMMU translation disabled.
- “DRAM” really means “Memory Controller”. In particular:
  - The GART range (5700_0000 – 5FFF_FFFF) has been used as an SMMU bounce range, which through the SMMU points to DRAM.
  - If the system configuration includes a PCIe device, then the PCIE_A* regions need to be configured as MMIO (appropriately as required by address space requirement). If the system does not include any PCIe devices, then these regions should be configured as DRAM.
  - All configurable regions that do not have a real MMIO target (labeled as DRAM+) in the above table should be configured as DRAM to provide the largest and uniform IOVA view (in 2GB or 4GB system) for the AVP and other clients that generate virtual addresses to access DRAM.
- Requests from all clients (except the main CPU) are allowed to use SMMU translation. Any request address range that can reach the MC can be translated by the SMMU.
- SMMU is a sub-unit in the memory controller. Address ranges marked as "No" in the table will not reach the memory controller.
- All Host1x clients can see the full 4GB physical or IOVA range.

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

Table 3: Client to SW Name Mapping

SW Group	Client	Description
AFI	AFI	PCIe reads
AVPC	AVP Cache	ARM7 Audio-Video Processor (AVP)
DC	Display	Display reads, window A, window C, window D, window T, cursor
DCB	Displayb	Display reads, window B, window C, cursor
GPU	GPU	Kepler GPU, 3D engine
HC	Host1x	Host interface
HDA	HDA	High-definition Audio engine
ISP2	ISP2	Image Signal Processor, instance a
ISP2B	ISP2B	Image Signal Processor, instance b
MPCORE	AXICIF	Cortex-A15 CPU cores, writes
MPCORELP	AXICIF_LP	Cortex-A15 Shadow CPU core, writes
MSENC	MSENC	Multi Standard Video Encoder
PPCS	AHBDMA, AHBSLV	Clients in the AHB cluster (ppcs2mc_swid=1'b0)
PPCS1	AHBDMA, AHBSLV	Same as PPCS, ppcs2mc_swid=1'b0)
PPCS2	AHBDMA, AHBSLV	Same as PPCS, ppcs2mc_swid=1'b0)
PTC	MC (SMMU)	Misses from SMMU PTC
SATA	SATA	Serial ATA
SDMMC1a	SDMMCA	SDMMC1
SDMMC2a	SDMMCAA	SDMMC2
SDMMC3a	SDMMC	SDMMC3
SDMMC4a	SDMMCAB	SDMMC4
TSEC	TSEC	Tegra Security coprocessor
VDE	VDE	Video Decode Engine
VI	VI2	Video Input engine for CSI
VIC	VIC	Video Image Compositer
XUSB_HOST	USB3 Host	Although the same driver is expected to control both units, each has its own SWNAME
XUSB_DEV	USB3 Device	

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

The clients behind AHBSLV in the above table are:

- VCP
- CSITE
- ARC
- AHBDMA
- USB
- APBDMA
- SNOR
- BSEV
- SE
- DDS
- BSEA
- USB2
- MIPIHSI

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

3.0 INTERRUPT CONTROLLER

This section provides the mapping and hierarchical groups of the Tegra® K1 family processor interrupts. It also describes the architecture for routing and handling of these interrupts. It describes semaphores, which provide a mechanism for the processors to arbitrate for the use of various resources.

From the perspective of interrupts: devices (GPU, Memory Controller, Video encode/decode engines, and various I/O devices) are the sources of interrupts; and processors are the target of interrupts. From sources, interrupts go to interrupt controllers which, based on configuration, prioritize and route them to the appropriate target processor. There are two different types of interrupt controllers used in Tegra K1 devices: The ARM® vGIC and the Legacy Interrupt Controller (LIC).

Glossary and Acronyms

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

3.1 References

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

Document	Description
vGIC Architecture	ARM Virtual Generic Interrupt Controller (also known as GIC v2) Architectural Specification
Timers Architecture	ARM Generic Timer Architectural Specification
Cortex-A15 TRM	Cortex-A15 Technical Reference Manual for Cortex-A15 implementation specific information (Tegra K1 32-bit).
ARM Technical Reference Manual v8	For implementation specific information (Tegra K1 64-bit)

3.2 Interrupt Mapping

The following five tables show the mapping of the interrupts from system devices to the bit fields in the Tegra K1 interrupt controllers.

Interrupts to the CPU's embedded interrupt controller (vGIC) are in this same order, but start at offset 32 because the first 32 are reserved for the CPU's internal interrupts.

Table 4: Primary Interrupt Controller (PRI_ICTRL) Mapping

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

Table 5: Secondary Interrupt Controller (SEC_ICTLR) Mapping

Global Interrupt Number	Interrupt Target	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description	Interrupt Order
63		31	I2C6	I2C	I2C6 Interrupt	0
62		30	CLDVFS	CL-DVFS	Close Loop DVFS control logic	0
61	COP	29	AHB_DMA_COP	AHB_DMA	AHB-DMA Interrupt (COP)	1
60	COP	28	APB_DMA_COP	APB_DMA	APB-DMA Interrupt (COP)	1
59		27	SPI1	SPI	SPI Controller (SBC1)	0
58		26	SE	SE	SE and TZRAM General Interrupt	0
57		25	USB3_DEV_PME	USB3	USB3_DEV_PME	0

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

Table 6: Tertiary Interrupt Controller (TRI_ICTLR) Mapping

Global Interrupt Number	Interrupt Target	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description	Interrupt Order
95		31	SW_INTR	N/A (Software)	SW Controlled Interrupt	0
94		30	Unmapped		Unassigned	0
93		29	Unmapped		Unassigned	0
92		28	I2C3	I2C	I2C3 Interrupt	0
91		27	Unmapped		Unassigned	0
90		26	UARTD	UART	UART-D	0
89		25	GPIO7	GPIO (external)	GPIO7 Interrupt	0
88		24	Unmapped		Unassigned	0
87		23	GPIO6	GPIO (external)	GPIO6 Interrupt	0
86		22	PMU_EXT	PMIC	External Power Management Chip Interrupt	0
85		21	Unmapped		Unassigned	0
84		20	I2C2	I2C	I2C2 Interrupt	0
83		19	SPI3	SPI	SPI Controller (SBC3)	0

Global Interrupt Number	Interrupt Target	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description	Interrupt Order
82		18	SPI2	SPI	SPI Controller (SBC2)	0
81		17	HDA	HDA	HDA Interrupt	0
80		16	Unmapped		Unassigned	0
79		15	SPI6	SPI	SPI Controller	0
78		14	EMC	EMC	EMC General Interrupt	0
77		13	MC	MC	MC General Interrupt	0
76		12	SOR	SOR (Display)	SOR (eDP/LVDS) General Interrupt	0
75		11	HDMI	HDMI	HDMI INT from pins	0
74		10	DISPLAYB	DISPLAYB	Display B General Interrupt	0
73		9	DISPLAY	DISPLAY	Display A General Interrupt	0
72		8	VIC	VIC	Video Image Compositer General Interrupt	0
71		7	ISP	ISP	ISP General Interrupt	0
70		6	ISPB	ISPB	ISPB General Interrupt	0
69		5	VI	VI	VI General Interrupt	0
68		4	MSENC	MSENC	MSENC General Interrupt	0
67	CPU	3	HOST1X_GEN_CPU	HOST1X	CPU General Interrupt	3
66	COP1	2	HOST1X_GEN_COP	HOST1X	COP General Interrupt	2
65	CPU	1	HOST1X_SYNCPT_CPU	HOST1X	CPU SyncPt Interrupt	1
64	COP1	0	HOST1X_SYNCPT_COP	HOST1X	COP SyncPt Interrupt	0

Table 7: Quaternary Interrupt Controller (QUAD_CTLR) Mapping

Global Interrupt Number	Interrupt Target	LIC Interrupt Number	Interrupt Name	Source Block	Description	Interrupt Order
127		31	HIER_GROUP1_CPU	HIER_GROUP1	CPU interrupt from hierarchical group1.	0
126		30	CAR	CAR	CAR Interrupt	0
125		29	GPIO8	GPIO (external)	GPIO8 Interrupt	0
124		28	WDT_AVP	Timer	Watchdog AVP Interrupt	2
123		27	WDT_CPU	Timer	Watchdog CPU Interrupt	1
122		26	HIER_GROUP1_COP	HIER_GROUP1	COP interrupt from hierarchical group1.	0
121		25	TMR5	Timer	TMR5 Interrupt	0
120		24	I2C4	I2C	I2C4 Interrupt	0
119		23	APB_DMA_CH15	APB_DMA	APB-DMA Channel 15	0
118		22	APB_DMA_CH14	APB_DMA	APB-DMA Channel 14	0
117		21	APB_DMA_CH13	APB_DMA	APB-DMA Channel 13	0
116		20	APB_DMA_CH12	APB_DMA	APB-DMA Channel 12	0
115		19	APB_DMA_CH11	APB_DMA	APB-DMA Channel 11	0
114		18	APB_DMA_CH10	APB_DMA	APB-DMA Channel 10	0
113		17	APB_DMA_CH9	APB_DMA	APB-DMA Channel 9	0
112		16	APB_DMA_CH8	APB_DMA	APB-DMA Channel 8	0
111		15	APB_DMA_CH7	APB_DMA	APB-DMA Channel 7	0

Global Interrupt Number	Interrupt Target	LIC Interrupt Number	Interrupt Name	Source Block	Description	Interrupt Order
110		14	APB_DMA_CH6	APB_DMA	APB-DMA Channel 6	0
109		13	APB_DMA_CH5	APB_DMA	APB-DMA Channel 5	0
108		12	APB_DMA_CH4	APB_DMA	APB-DMA Channel 4	0
107		11	APB_DMA_CH3	APB_DMA	APB-DMA Channel 3	0
106		10	APB_DMA_CH2	APB_DMA	APB-DMA Channel 2	0
105		9	APB_DMA_CH1	APB_DMA	APB-DMA Channel 1	0
104		8	APB_DMA_CH0	APB_DMA	APB-DMA Channel 0	0
103		7	AUDIO_CLUSTER	AUDIO_ CLUSTER	Audio Cluster Interrupt	0
102		6	Unmapped		Unassigned	0
101		5	AVP_CACHE	ARM7 Cache	ARM7 Cache Interrupt	0
100		4	PCIE_WAKE	PCIe	PCIe wake-up signal	0
99		3	PCIE_MSI	PCIe	PCIe message signaled interrupt	0
98		2	PCIE_INT	PCIe	PCIe catch-all error/legacy interrupts	0
97		1	USB3	USB	USB Device	0
96		0	SNOR	SNOR	Sync NOR Controller	0

Table 8: Fifth Interrupt Controller (PENTA_ICTLR) Mapping

Global Interrupt Number	Interrupt Target	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description	Interrupt Order
159		31	ARDPAUX	DPAUX	ARDPAUX General Interrupt	0
158		30	GPU_NONSTALL_ INT	GPU	GPU Interrupt	0
157		29	GPU_INT	GPU	GPU Interrupt	0
156		28	TMR0	Timer	TMR0 Interrupt	0
155		27	TMR9	Timer	TMR9 Interrupt	0
154		26	TMR8	Timer	TMR8 Interrupt	0
153		25	TMR7	Timer	TMR7 Interrupt	0
152		24	TMR6	Timer	TMR6 Interrupt	0
151		23	SDMMC4_SYS	SDMMC	SDMMC4 interrupt for standard driver (e.g., WOA)	0
150		22	SDMMC3_SYS	SDMMC	SDMMC3 interrupt for standard driver (e.g., WOA)	0
149		21	SDMMC2_SYS	SDMMC	SDMMC2 interrupt for standard driver (e.g., WOA)	0
148		20	SDMMC1_SYS	SDMMC	SDMMC1 interrupt for standard driver (e.g., WOA)	0
147		19	CPU3_PMU_INTR	A15 CPU3	CPU3 Performance Management Unit (PMU) Interrupt	0
146		18	CPU2_PMU_INTR	A15 CPU2	CPU2 PMU Interrupt	0
145		17	CPU1_PMU_INTR	A15 CPU1	CPU1 PMU Interrupt	0
144		16	CPU0_PMU_INTR	A15 CPU0	CPU0 PMU Interrupt	0
143		15	APB_DMA_CH31	APB_DMA	APB-DMA Channel 31	0
142		14	APB_DMA_CH30	APB_DMA	APB-DMA Channel 30	0
141		13	APB_DMA_CH29	APB_DMA	APB-DMA Channel 29	0
140		12	APB_DMA_CH28	APB_DMA	APB-DMA Channel 28	0
139		11	APB_DMA_CH27	APB_DMA	APB-DMA Channel 27	0

Global Interrupt Number	Interrupt Target	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description	Interrupt Order
138		10	APB_DMA_CH26	APB_DMA	APB-DMA Channel 26	0
137		9	APB_DMA_CH25	APB_DMA	APB-DMA Channel 25	0
136		8	APB_DMA_CH24	APB_DMA	APB-DMA Channel 24	0
135		7	APB_DMA_CH23	APB_DMA	APB-DMA Channel 23	0
134		6	APB_DMA_CH22	APB_DMA	APB-DMA Channel 22	0
133		5	APB_DMA_CH21	APB_DMA	APB-DMA Channel 21	0
132		4	APB_DMA_CH20	APB_DMA	APB-DMA Channel 20	0
131		3	APB_DMA_CH19	APB_DMA	APB-DMA Channel 19	0
130		2	APB_DMA_CH18	APB_DMA	APB-DMA Channel 18	0
129		1	APB_DMA_CH17	APB_DMA	APB-DMA Channel 17	0
128		0	APB_DMA_CH16	APB_DMA	APB-DMA Channel 16	0

3.3 Hierarchical Groups

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

Table 9: Hierarchical Interrupt Grouping

Hierarchical Interrupt Controller Name	Interrupt Number	Interrupt Name	Source Block	Interrupt Description
HIER_GROUP1	31 - 16	Currently Unmapped		Currently Unassigned
HIER_GROUP1	15	MselectError		Unassigned
HIER_GROUP1	14	MPCORE_CTIIRQ3		CPU3 CTI interrupt
HIER_GROUP1	13	MPCORE_CTIIRQ2		CPU2 CTI interrupt
HIER_GROUP1	12	MPCORE_CTIIRQ1		CPU1 CTI interrupt
HIER_GROUP1	11	MPCORE_CTIIRQ0		CPU0 CTI interrupt
HIER_GROUP1	10	TMR_SHARED	Timer	TMR_SHARED interrupt
HIER_GROUP1	9	FLOW_RSM_COP	FlowCtlr	FLOW - RSM1 Resume for COP or COP1
HIER_GROUP1	8	FLOW_RSM_CPU	FlowCtlr	FLOW - RSM0 Resume for CPU
HIER_GROUP1	7	Currently Unmapped		Currently Unassigned
HIER_GROUP1	6	Currently Unmapped		Currently Unassigned
HIER_GROUP1	5	EVENT_GPIO_D	GPIO (External)	Event Detector GPIO Port D
HIER_GROUP1	4	EVENT_GPIO_C	GPIO (External)	Event Detector GPIO Port C
HIER_GROUP1	3	EVENT_GPIO_B	GPIO (External)	Event Detector GPIO Port B
HIER_GROUP1	2	EVENT_GPIO_A	GPIO (External)	Event Detector GPIO Port A
HIER_GROUP1	1	MPCORE_INTERRIRQ	CCPLEX	CPU INTERRIRQ
HIER_GROUP1	0	MPCORE_AXIERRIRQ	CCPLEX	CPU AXIERRIRQ

3.4 Functional Description

3.4.1 Interrupt Handling Mechanism

The two different types of interrupt controllers (vGIC, and LIC) receive interrupts from devices (i.e., hardware interrupts also known as SPIs) or processors (i.e., software interrupts including IPI), arbitrate them, and send them to the appropriate target

processor(s). From an interrupt perspective, there are two different types of processors: CPUs (Cortex-A15) and COP (ARM7 AVP). The vGIC is the interrupt controller for the CPUs, and the LIC is the interrupt controller for the ARM7 AVP. Any processor can initiate a software interrupt, targeted to any one or more processors (including itself). However, IPIs can only be initiated by a Cortex-A15 CPU to any one or more Cortex-A15 CPUs (including itself).

There are 160 hardware interrupts in Tegra K1 devices. Interrupt sources are allocated one or more interrupts as required. The 160 interrupts are grouped into slices of 32, where each slice can be configured independently.

3.4.1.1 ARM Processors' IRQ and FIQ

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

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

3.4.1.2 Interrupt Controllers

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

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

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

The interrupt controller used for CPUs (Cortex-A15 CPUs) is called vGIC (virtual generic interrupt controller). The vGIC is an SMP interrupt controller. It receives interrupts targeted to any one or more of the CPUs in the SMP complex. There is one vGIC per CPU cluster.

3.4.1.3 vGIC Interrupt Sources

The vGIC supports 256 interrupts. All interrupts are identified by a unique ID. There are three types of interrupt sources for the vGIC: SGIs, PPIs, and SPIs.

Software Generated Interrupts (SGIs)

SGIs are software interrupts which are generated by writing to a vGIC register. Software can generate a maximum of 16 SGIs, with ID0-15. The SGIs are also referred to as IPIs (Inter Processor Interrupts).

Private Peripheral Interrupts (PPIs)

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

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

Tegra K1 devices use all PPIs except for the legacy nFIQ and legacy nIRQ.

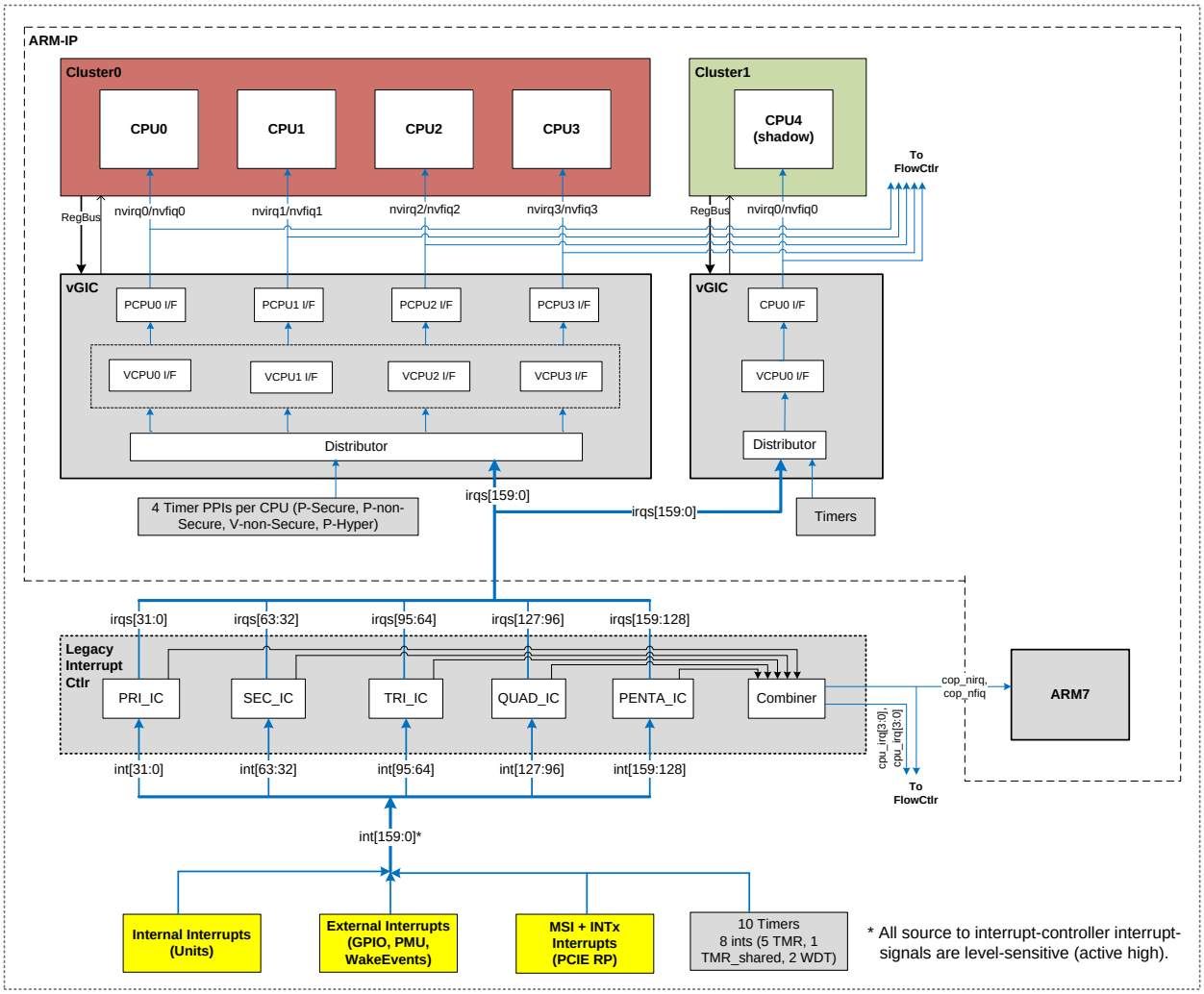
Shared Peripheral Interrupts (SPIs)

SPIs are external hardware interrupts which are generated by asserting signals on vGIC input pins (called IRQs). From a Tegra hardware perspective, these are the interrupts which are generated by various units to be sent to interrupt controllers. Tegra K1 devices use 160 of the possible 224 SPIs generated through the IRQS[223:0] pins. SPIs start at ID32. Tegra K1 devices use level-sensitive SPIs only.

3.4.1.4 Interrupt Routing

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

Figure 2: Interrupt Routing (Top-Level View) for Tegra K1 32-Bit



As shown in the figure above, hardware interrupts are routed to the interrupt controllers (called IC32) within the LIC, where each IC32 handles 32 interrupts. The interrupt signals are synchronized within the LIC to the system clock. The synchronized signals are combined with software set/clear bits per interrupt. This allows software to set or clear individual interrupts,

provided the corresponding interrupt is not asserted by hardware. The resulting interrupt signals are broadcast to the LIC (for further processing) and the vGIC. Within the LIC and vGIC, per interrupt and per CPU enable-masks are configured to qualify the interrupt targeted for the corresponding processor. This mechanism allows any interrupt to be targeted to any one or more processors in Tegra K1 devices.

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

3.4.1.5 Interrupt Handling with Targeted CPU and vGIC Powered Off

When a CPU and associated vGIC are powered off (e.g., the CPU rail is powered off for cluster0), then the legacy interrupt controller is configured to act as a “proxy” interrupt controller to generate IRQ/FIQ for that CPU. Also, Tegra K1 timers (instead of vGIC timers) have to be set up if a timer interrupt is required. The flow controller uses legacy interrupt controller IRQ/FIQ to power-up the targeted CPU (and vGIC).

The following sequence describes the steps to follow to handle interrupts when the targeted CPU and vGIC are powered off:

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

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

3.4.1.6 Interrupt Handling with Targeted CPU Powered Off

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

The following sequence describes the steps to follow to handle interrupts when the targeted CPU is powered off:

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

Because the vGIC IRQ and FIQ are level-triggered, the interrupt that triggered the wake-up is still asserted.

3.4.2 Virtual Generic Interrupt Controller (Tegra K1 32-Bit)

This section describes the Cortex-A15 vGIC (also known as GICv2). Refer to the ARM Cortex-A15 TRM for the vGIC implementation specific specification and/or the ARM vGIC architectural specification.

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

3.4.2.1 Functional Description

Split EOI Functionality

Every interrupt has a priority, and any interrupt being active will prevent another interrupt of equal (or lower) priority from being asserted to the CPU. This causes problems for virtualization where there is not necessarily any well-defined priority for interrupts between the various VMs. To address this problem the functionality associated with EOI is divided into two distinct functions:

- Priority Drop to be used after the interrupt is acknowledged to drop the running priority of the CPU back down but leave the interrupt in the ACTIVE state
- Deactivate Interrupt to be used when the interrupt handling is truly completed by the VM, causing the interrupt to transition from ACTIVE to INACTIVE in the physical distributor.

To avoid Hypervisor intervention being necessary when a Guest OS completes interrupt processing, the virtual CPU Interface is able to directly trigger a Deactivate Interrupt event on the interrupt distributor.

Virtual CPU Interface

The Virtual CPU Interface supports CPU virtualization.

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

Security and Virtual FIQ

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

Interrupt Processing in the vGIC

- Physical Interrupt Asserted

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

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

- Virtual Interrupt Asserted

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

- Virtual Interrupt Acknowledged

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

- Virtual Interrupt EOI

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

Timers

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

- A Secure Physical Timer and a Non-Secure Physical Timer for use in Secure and Non-secure PL1 modes, respectively. These are generated as two separate interrupts to the core: ID29 (secure) and ID30 (non-secure).
- Virtual Timer for use in Non-secure PL1 modes.
- Physical Timer for use in Hypervisor mode.

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

For more information, see the Timers section in this TRM and the *ARM Architecture Reference Manual v7* for generic timer specifications.

3.4.2.2 vGIC Clock Frequency

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

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

3.4.2.3 vGIC Context Save/Restore

The vGIC and timers are part of the Cortex-A15 non-CPU (also known as L2) logic. The cluster0 vGIC is powered by the CPU rail, and cluster1 is powered by the SoC rail. Also, if non-CPU is power gated then vGIC is power gated as part of non-CPU partition.

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

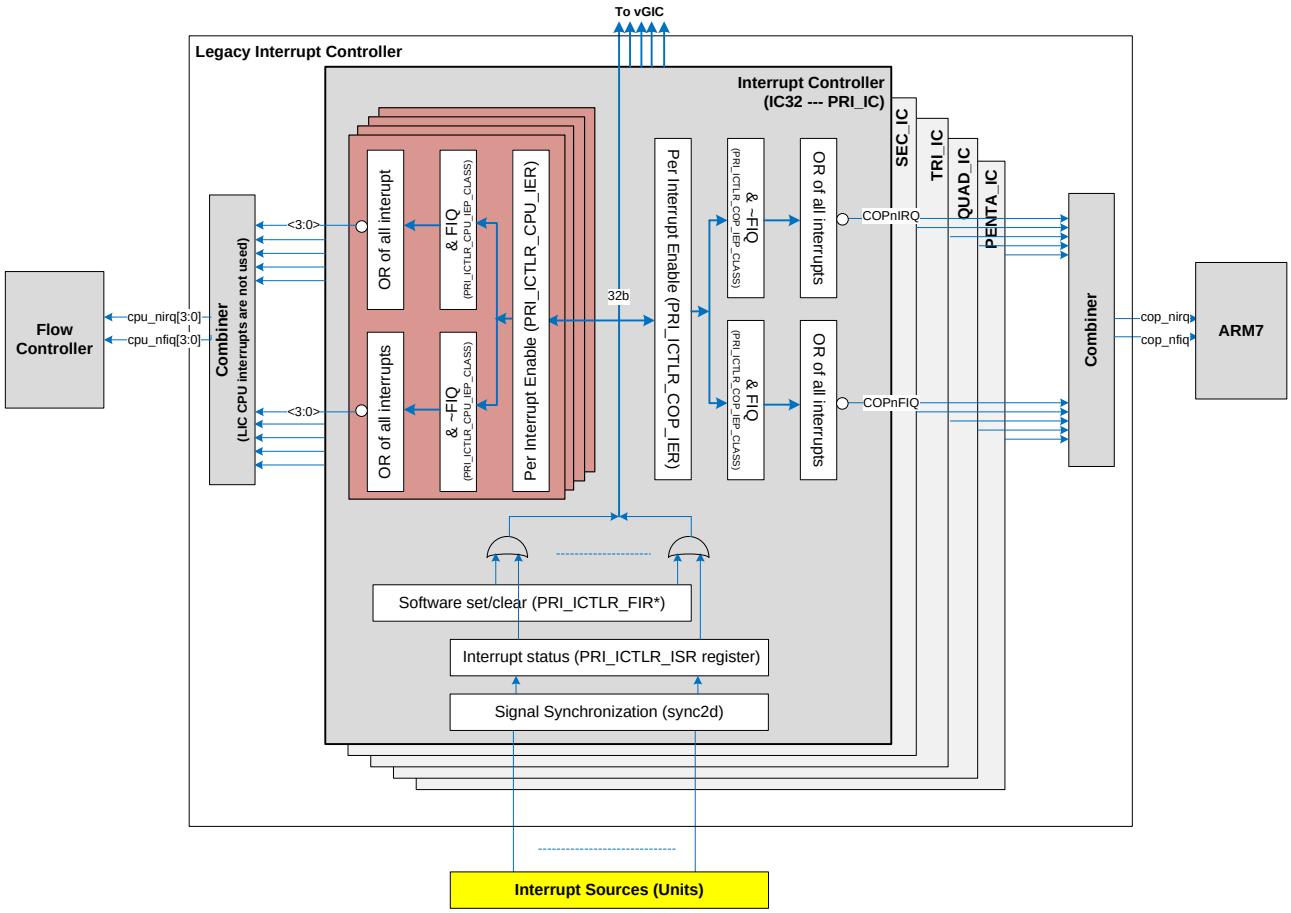
- When CPU(s) is individually power gated (i.e., LP2 power state), the vGIC is still powered, so the vGIC context does not have to be saved/restored. Therefore, CPUs can be individually powered-off (that is, put into the LP2 state) and powered on without saving/restoring their vGIC context.
- When the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

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

3.4.3 Legacy Interrupt Controller (LIC)

Interrupts from various Tegra K1 units are routed to the LIC. The LIC consists of five interrupt controllers (called IC32); each IC32 handles 32 interrupts (for a total of 160 interrupts). Each IC32 generates five sets of IRQ/FIQ interrupts: one for COP (ARM7) and the other ones for CPU<3:0>. The interrupts for CPU<3:0> are used by the flow controller only for wake events.

Figure 3: Legacy Interrupt Controller Block Diagram



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

3.4.3.1 LIC Functional Description

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

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

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

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

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

3.4.3.2 LIC Registers Description

The LIC consists of five interrupt controllers (called IC32); each IC32 handles 32 interrupts (for a total of 160 interrupts). Each IC32 defines two sets of registers: one set for the COP (ARM7) and other set for the Cortex-A15 CPU.

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

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

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

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

The differences between ISR and VFIQ/VIRQ registers are:

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

The VIRQ_COP and VIRQ_CPU registers are calculated as follows.

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

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

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

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

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

3.4.3.3 Hierarchical Interrupts Grouping

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

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

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

The HIER_GROUPx_{CPU,COP}_ENABLE registers act as mask registers for the hierarchical interrupts. Each bit in the ENABLE register can mask the interrupt for the corresponding hierarchical group. Refer to Table 9 for the hierarchical interrupt grouping per bit.

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

3.4.4 Arbitration Semaphores

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

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

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

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

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

3.4.5 Interrupts Used for Processor Power-Ups

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

3.5 Interrupt Registers

3.5.1 Interrupt Controller Registers

NOTE: The interrupt controller (IC32) register descriptions do not follow the standard register table protocol described in "Reading Register Tables" in the Introduction section. Each register's relative offset, access type, and power-on reset value are located in Table 11. The bit descriptions for each register are defined in Table 12 according to interrupt controller (IC32).

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

- ID identifies the specific IC32: PRI_ICTLR, SEC_ICTLR, TRI_ICTLR, QUAD_ICTLR, PENTA_ICTLR. Based on the IC32, ID defines a base address and mapping (see below).

- FUNCTION identifies the use for this register. Refer to the LIC Registers Description subsection for more details. FUNCTION is defined as one of the following:
  - VIRQ_<DEST> and VFIQ_<DEST>: indicates the Valid status, as an IRQ or FIQ, respectively, for this destination. DEST is COP, CPU, CPU1, CPU2, or CPU3.
  - ISR: indicates the latched Interrupt Status
  - FIR: indicates Force Interrupt
  - FIR_SET and FIR_CLR: to control FIR, bits set to 1 are set or cleared, respectively, in FIR.
  - <DEST>_IER: indicates that the corresponding event is enabled as an interrupt
  - <DEST>_IER_SET and <DEST>_IER_CLR: to control IER, bits equal to 1 are set or cleared, respectively, in IER.
  - <DEST>_IEP_CLASS: Interrupt Enable Priority Class, a bit set to 0 indicates IRQ, and set to 1 indicates FIQ.

Table 10 associates the following with each IC32: the ID (used in the register names), the base offset (address map of the five IC32s), and interrupt mapping reference.

Table 10: IC32 Characteristics

ID	Base Offset	Mapping
PRI_ICTLR	0x000	See Table 4: Primary Interrupt Controller (PRI_ICTLR) Mapping
SEC_ICTLR	0x100	See Table 5: Secondary Interrupt Controller (SEC_ICTLR) Mapping
TRI_ICTLR	0x200	See Table 6: Tertiary Interrupt Controller (TRI_ICTLR) Mapping
QUAD_ICTLR	0x300	See Table 7: Quaternary Interrupt Controller (QUAD_CTLR) Mapping
PENTA_ICTLR	0x400	See Table 8: Fifth Interrupt Controller (PENTA_ICTLR) Mapping
HIER2_ICTLR1	0x800	

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

Table 11: Interrupt Controller IC32 Register Set

Name	Relative Offset	Read/ Write	Reset	Remark
<ID>_VIRQ_CPU_0	0x00	RO	x	Valid Interrupt Request Status for CPU0 Register
<ID>_VIRQ_COP_0	0x04	RO	x	Valid Interrupt Request Status for COP Register
<ID>_VFIQ_CPU_0	0x08	RO	x	FIQ Valid Interrupt Status for CPU0 Register
<ID>_VFIQ_COP_0	0x0C	RO	x	FIQ Valid Interrupt Status for COP Register
<ID>_ISR_0	0x10	RO	x	Latched Interrupt Status Register
<ID>_FIR_0	0x14	RO	x	Forced Interrupt Status Register
<ID>_FIR_SET_0	0x18	WO	0	Force Interrupt Register Set Register
<ID>_FIR_CLR_0	0x1C	WO	0	Force Interrupt Register Clear Register
<ID>_CPU_IER_0	0x20	RO	x	Enabled Interrupt Source for CPU0 Register (0=disabled)
<ID>_CPU_IER_SET_0	0x24	WO	0	Set Interrupt Enable for CPU0 Register
<ID>_CPU_IER_CLR_0	0x28	WO	0	Clear Interrupt Enable for CPU Register
<ID>_CPU_IEP_CLASS_0	0x2C	RW	0	CPU0 Interrupt Enable Priority Class Register (1=FIQ/0=IRQ)

Name	Relative Offset	Read/ Write	Reset	Remark
<ID>_COP_IER_0	0x30	RO	x	Enabled Interrupt Source for COP Register (0=disabled
<ID>_COP_IER_SET_0	0x34	WO	0	Set Interrupt Enable for COP Register
<ID>_COP_IER_CLR_0	0x38	WO	0	Clear Interrupt Enable for COP Register
<ID>_COP_IEP_CLASS_0	0x3C	RW	0	COP Interrupt Enable Priority Class Register (1=FIQ/0=IRQ)
<ID>_VIRQ_CPU1_0	0x60	RO	x	Valid Interrupt Request Status for CPU1 Register
<ID>_VFIQ_CPU1_0	0x64	RO	x	FIQ Valid Interrupt Status for CPU Register
<ID>_CPU1_IER_0	0x68	RO	x	Enabled Interrupt Source for CPU1 Register (0=disabled)
<ID>_CPU1_IER_SET_0	0x6C	WO	0	Set Interrupt Enable for CPU1 Register
<ID>_CPU1_IER_CLR_0	0x70	WO	0	Clear Interrupt Enable for CPU1 Register
<ID>_CPU1_IEP_CLASS_0	0x74	RW	0	CPU1 Interrupt Enable Priority Class Register (1=FIQ/0=IRQ)
<ID>_VIRQ_CPU2_0	0x78	RO	x	Valid Interrupt Request Status for CPU2 Register
<ID>_VFIQ_CPU2_0	0x7C	RO	x	FIQ Valid Interrupt Status for CPU2 Register
<ID>_CPU2_IER_0	0x80	RO	x	Enabled Interrupt Source for CPU2 Register (0=disabled)
<ID>_CPU2_IER_SET_0	0x84	WO	0	Set Interrupt Enable for CPU2 Register
<ID>_CPU2_IER_CLR_0	0x88	WO	0	Clear Interrupt Enable for CPU2 Register
<ID>_CPU2_IEP_CLASS_0	0x8c	RW	0	CPU2 Interrupt Enable Priority Class Register (1=FIQ/0=IRQ)
<ID>_VIRQ_ CPU3_0	0x90	RO	x	Valid Interrupt Request Status for CPU3 Register
<ID>_VFIQ_ CPU3_0	0x94	RO	x	FIQ Valid Interrupt Status for CPU3 Register
<ID>_CPU3_IER_0	0x98	RO	x	Enabled Interrupt Source for CPU3 Register (0=disabled)
<ID>_CPU3_IER_SET_0	0x9C	WO	0	Set Interrupt Enable for CPU3 Register
<ID>_CPU3_IER_CLR_0	0xA0	WO	0	Clear Interrupt Enable for CPU3 Register
<ID>_CPU3_IEP_CLASS_0	0xA4	RW	0	CPU3 Interrupt Enable Priority Class Register (1=FIQ/0=IRQ)

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

Table 12: IC32 Register Bit Descriptions

Bit	First IC32 (PRI_ICTLR) Bits (Base Interrupt = 0)	Second IC32 (SEC_ICTLR) Bits (Base Interrupt = 32)	Third IC32 (TRI_ICTLR) Bits (Base Interrupt = 64)	Fourth IC32 (QUAD_ICTRL) Bits (Base Interrupt = 96)	Fifth IC32 (PENTA_ICTRL) Bits (Base Interrupt = 128)
31	SDMMC4	I2C6	SW_INTR	HIER_GROUP1_CPU	DPAUX
30	OWR	CLDVFS	SPI5	CAR	GPU_NONSTALL
29	ARB_SEM_GNT_CPU	AHB_DMA_COP	SPI4	GPIO8	GPU_STALL
28	ARB_SEM_GNT_COP	APB_DMA_COP	I2C3	WDT_AVP	TMR0
27	AHB_DMA_CPU	SPI1	-	WDT_CPU	TMR9
26	APB_DMA_CPU	SE	UARTD	HIER_GROUP1_COP	TMR8
25	VCP	USB3_DEV_PME	GPIO7	TMR5	TMR7
24	-	USB3_DEV_SMI	-	I2C4	TMR6
23	SATA_CTL	GPIO5	GPIO6	APB_DMA_CH15	SDMMC4_SYS
22	-	STAT_MON	PMU_EXT	APB_DMA_CH14	SDMMC3_SYS
21	USB2	I2C5	-	APB_DMA_CH13	SDMMC2_SYS
20	USB	VFIR	I2C2	APB_DMA_CH12	SDMMC1_SYS
19	SDMMC3	EDP	SPI3	APB_DMA_CH11	CPU3_PMU_INTR
18	AVP_UCQ	TSEC	SPI2	APB_DMA_CH10	CPU2_PMU_INTR

Bit	First IC32 (PRI_ICTLR) Bits (Base Interrupt = 0)	Second IC32 (SEC_ICTLR) Bits (Base Interrupt = 32)	Third IC32 (TRI_ICTLR) Bits (Base Interrupt = 64)	Fourth IC32 (QUAD_ICTRL) Bits (Base Interrupt = 96)	Fifth IC32 (PENTA_ICTRL) Bits (Base Interrupt = 128)
17	VDE	XUSB_PADCTL	HDA	APB_DMA_CH9	CPU1_PMU_INTR
16	-	THERMAL	-	APB_DMA_CH8	CPU0_PMU_INTR
15	SDMMC2	HSI	SPI6	APB_DMA_CH7	APB_DMA_CH31
14	SDMMC1	UARTC	EMC	APB_DMA_CH6	APB_DMA_CH30
13	SATA_RX_STAT	ACTMON	MC	APB_DMA_CH5	APB_DMA_CH29
12	VDE_SXE	USB3_DEV_HOST	SOR	APB_DMA_CH4	APB_DMA_CH28
11	VDE_BSEA	USB3_HOST_PME	HDMI	APB_DMA_CH3	APB_DMA_CH27
10	VDE_BSEV	TMR4	DISPLAYB	APB_DMA_CH2	APB_DMA_CH26
9	VDE_SYNC_TOKEN	TMR3	DISPLAY	APB_DMA_CH1	APB_DMA_CH25
8	VDE_UCQ	USB3_HOST_SMI	VIC	APB_DMA_CH0	APB_DMA_CH24
7	SHR_SEM_OUTBOX_EMPTY	USB3_HOST_INT	ISP	AUDIO_CLUSTER	APB_DMA_CH23
6	SHR_SEM_OUTBOX_FULL	I2C	ISPB	-	APB_DMA_CH22
5	SHR_SEM_INBOX_EMPTY	UARTB	VI	AVP_CACHE	APB_DMA_CH21
4	SHR_SEM_INBOX_FULL	UARTA	MSENC	PCIE_WAKE	APB_DMA_CH20
3	CEC	GPIO4	HOST1X_GEN_CPU	PCIE_MSI	APB_DMA_CH19
2	RTC	GPIO3	HOST1X_GEN_COP	PCIE_INT	APB_DMA_CH18
1	TMR2	GPIO2	HOST1X_SYNCPT_CPU	USB3	APB_DMA_CH17
0	TMR1	GPIO1	HOST1X_SYNCPT_COP	SNOR	APB_DMA_CH16

3.5.2 Arb_gnt Specific Interrupt Controller Registers

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

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

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

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

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

3.5.2.1 PRI_ICTLR_ARBGNT_CPU_STATUS_0

CPU Arbitration Semaphore Interrupt Status Register

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

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

3.5.2.2 PRI_ICTLR_ARBGNT_CPU_ENABLE_0

CPU Arbitration Semaphore Interrupt Enable Register

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

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

3.5.2.3 PRI_ICTLR_ARBGNT_COP_STATUS_0

COP Arbitration Semaphore Interrupt Status Register

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

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

3.5.2.4 PRI_ICTLR_ARBGNT_COP_ENABLE_0

COP Arbitration Semaphore Interrupt Enable Register

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

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

3.5.3 Hier Group Interrupt Registers

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

This subsection describes the registers for Hier group interrupts for the CPU and the COP.

3.5.3.1 HIER_GROUP_CPU_ENABLE_0

Hier Group Enable for CPU

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

Bit	Reset	Description
31:0	0x0	CPU_ENABLE: Enable for the hier group interrupt for the CPU. Writing a 1 in any bit position will enable the corresponding interrupt.

3.5.3.2 HIER_GROUP_CPU_STATUS_0

Hier Group Status (ISR) Register for CPU

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

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

3.5.3.3 HIER_GROUP_COP_ENABLE_0

Hier Group Enable for COP

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

Bit	Reset	Description
31:0	0x0	COP_ENABLE: Enable for the hier group interrupt for the COP. Writing a 1 in any bit position will enable the corresponding interrupt.

3.5.3.4 HIER_GROUP_COP_STATUS_0

Hier Group Status (ISR) for COP

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

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

3.5.3.5 HIER_GROUP_FIR_STATUS_0

Status Register for FIR Register in Hier Group

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

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

3.5.3.6 HIER_GROUP_FIR_SET_0

FIR Set for Hier Group

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

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

3.5.3.7 HIER_GROUP_FIR_CLEAR_0

FIR Clear for Hier Group

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

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

3.5.4 EVP Registers

Each processor sees its own set of EVP reset vector registers.

3.5.4.1 EVP_RESET_VECTOR_0

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

Bit	Reset	Description
31:0	X	RESET_VECTOR: RESET Exception Vector Pointer

[THIS PAGE INTENTIONALLY LEFT BLANK]

4.0 SEMAPHORES

4.1 Arbitration Semaphores

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

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

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

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

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

4.2 Semaphore Registers

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

4.2.1 Arbitration Semaphore Registers

4.2.1.1 ARB_SEMA_SMP_GNT_ST_0

Semaphore Granted Status Register

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

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

4.2.1.2 ARB_SEMA_SMP_GET_0

Request Arbitration Semaphore Register

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

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

4.2.1.3 ARB_SEMA_SMP_PUT_0

Arbitration Semaphore Put Request Register

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

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

4.2.1.4 ARB_SEMA_SMP_REQ_ST_0

Arbitration Request Pending Status (1=PENDING) Register

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

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

4.2.2 Shared Resource Semaphore Registers

4.2.2.1 RES_SEMA_SHRD_SMP_STA_0

Shared Resource Semaphore Status

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

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

4.2.2.2 RES_SEMA_SHRD_SMP_SET_0

Shared Resource Semaphore Set-Bit Request

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

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

4.2.2.3 RES_SEMA_SHRD_SMP_CLR_0

Shared Resource Semaphore Clr-bit Request Register

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

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

4.2.2.4 RES_SEMA_SHRD_INBOX_0

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

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0b000x00000000000000000000000000000)

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

4.2.2.5 RES_SEMA_SHRD_OUTBOX_0

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

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0b000x00000000000000000000000000000)

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

5.0 CLOCK AND RESET CONTROLLER

The Clock and Reset (CAR) block contains all the logic needed to control most of the clocks and resets to the Tegra® K1 device. The CAR block provides the registers to program the PLLs and controls most of the clock source programming, and most of the clock dividers.

Power on reset (SYS_RESET_N_) is the primary reset for the Tegra K1 chip and is provided by the external PMC.

5.1 Hardware Features

5.1.1 External Clock Sources

The CAR block takes two external clock sources as input:

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

5.1.2 Clock Sources and PLLs

Clock sources are provided by 16 PLLs in the Tegra K1 clock system:

- PLLM: Clock source for EMC 2x clock
- PLLX: Clock source for the fast CPU cluster and the shadow CPU
- PLLC: Clock source for general use
- PLLC2 and PLLC3: Clock source for engine scaling
- PLLC4: Clock source for ISP/VI units
- PLLP: Clock source for most peripherals
- PLLA: Audio clock sources:
  - 11.2896 MHz
  - 12.288 MHz
  - 24.576 MHz
- PLLU: Clock source for USB PHY, provides 12/60/480 MHz
- PLLD and PLLD2: Clock sources for the DSI and display subsystem
- refPLLe and PLLE generate the 100 MHz reference clock for USB 3.0 and can generate a spread-spectrum clock
- PLLDP: Clock source for eDP/LVDS (spread spectrum)
- DFLLCPU: DFLL clock source for the fast CPU cluster
- GPCPLL: Clock source for the GPU

5.1.3 Clock Dividers / Skippers / Multiplier

There are a number of clock skippers and clock dividers as well as one clock multiplier in use in the Tegra K1 devices.

- Clock skippers:
  - An M/N clock-skipping divider that is used to generate the CPU clock and also the Tegra K1 "system clock" (sclk).
  - Clock skippers that are used to generate the "AHB clock" (hclk) and "APB clock" (pclk). One that divides down by a 2-bit unsigned value (U2).

- Clock dividers:
  - An unsigned 16-bit divider (U16) that is the divider for the I2C and UART devices.
  - An unsigned 8-bit divider that provides 7 bits of mantissa and 1 bit of fraction (U7.1). Tegra K1 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 K1 devices.

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

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

5.2 Clocking Architecture

Tegra K1 clocks are classified into two categories: core clocks and I/O clocks. Core clocks drive the non-I/O modules on the Tegra K1 chip. I/O clocks are custom designed for each I/O module. Only core clocks are described here.

5.2.1 Core Clocks

The core clock structure consists of:

- Sources
- Switches
- Distribution and skew balancing
- Second-level non-functional clock gates

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

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

5.2.2 Main Clock Sources

There are three types of clock sources:

- External clock sources
- PLLs
- Derived clock sources

Table 13: External Clocks to Tegra K1 Devices

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

Table 14: Primary Clock Sources in Tegra K1 Devices

Clock Name	Description
osc_div_clk:	This clock runs at 12 MHz, 13 MHz, 16.8 MHz, 19.2 MHz, or 26 MHz. Only 12 MHz is currently supported.
car_sclk:	This is the system clock whose maximum frequency is 200 MHz
clk_m:	This clock (with DFT control) runs at 12 MHz, 13 MHz, 16.8 MHz, 19.2 MHz, 26 MHz, 38.4 MHz, or 48 MHz. Only 12 MHz is currently supported.
car_clk_m:	This clock (with DFT control) runs at 12 MHz, 13 MHz, 16.8 MHz, 19.2 MHz, 26 MHz, 38.4 MHz, or 48 MHz. Only 12 MHz is currently supported.
dfllCPU_out:	This is the dfllCPU's output clock.
pllC_out:	This is the PLLC's output clock.
pllC2_out:	This is the PLLC2's output clock.
pllC3_out:	This is the PLLC3's output clock.
pllC4_out:	This is the PLLC4's output clock, which is used for the VI and ISP blocks.
pllM_out:	This is the PLLM's output clock.
int_pllA_out	This is the PLLA's output clock.
pllD_out:	This is the PLLD's output clock.
pllD2_out:	This is the PLLD2's output clock.
pllDP_out	This is the PLLDP's output clock.
pllU:	This is the PLLU's output clock. USB2 I/O.
refPLLE_out:	This is the refPLLE's output clock. Low jitter reference clock for PLLE, USB.
pllE_out0:	This is the PLLE's output clock. USB3 I/O.
pllP_out:	This is the PLLP's output clock which is set to 408 MHz.
pllX	This is the output clock for the fast CPU cluster and the shadow CPU
GPCPLL	This is the output clock for the GPU
clk_s:	This clock (with DFT control) runs at 32.768 kHz.
car_clk_s:	CAR's clock tree balanced version of clk_s.

Table 15: Derived Clock Sources in Tegra K1 Devices

Clock Name	Description
pllC_out1:	This is the U7.1 divider output clocked by "pllC_out".
pllM_out1:	This is the U7.1divider output clocked by "pllM_out".
pllP_out1:	This is the U7.1 divider output clocked by "pllP_out" (28.8 MHz boot, changed to 9.6 MHz).
pllP_out2:	This is the U7.1 divider output clocked by "pllP_out" (48 MHz boot, changed to 48 MHz).
pllP_out3:	This is the U7.1 divider output clocked by "pllP_out". (72 MHz boot, changed to 102 MHz).
pllP_out4:	This is the U7.1 divider output clocked by "pllP_out" (108 MHz boot, changed to variable)
pllP_out5:	This is the U7.1 divider output clocked by "pllP_out" (204 MHz boot, changed to variable).
pllA_out0:	This is the U7.1 divider output clocked by "int_pllA_out".

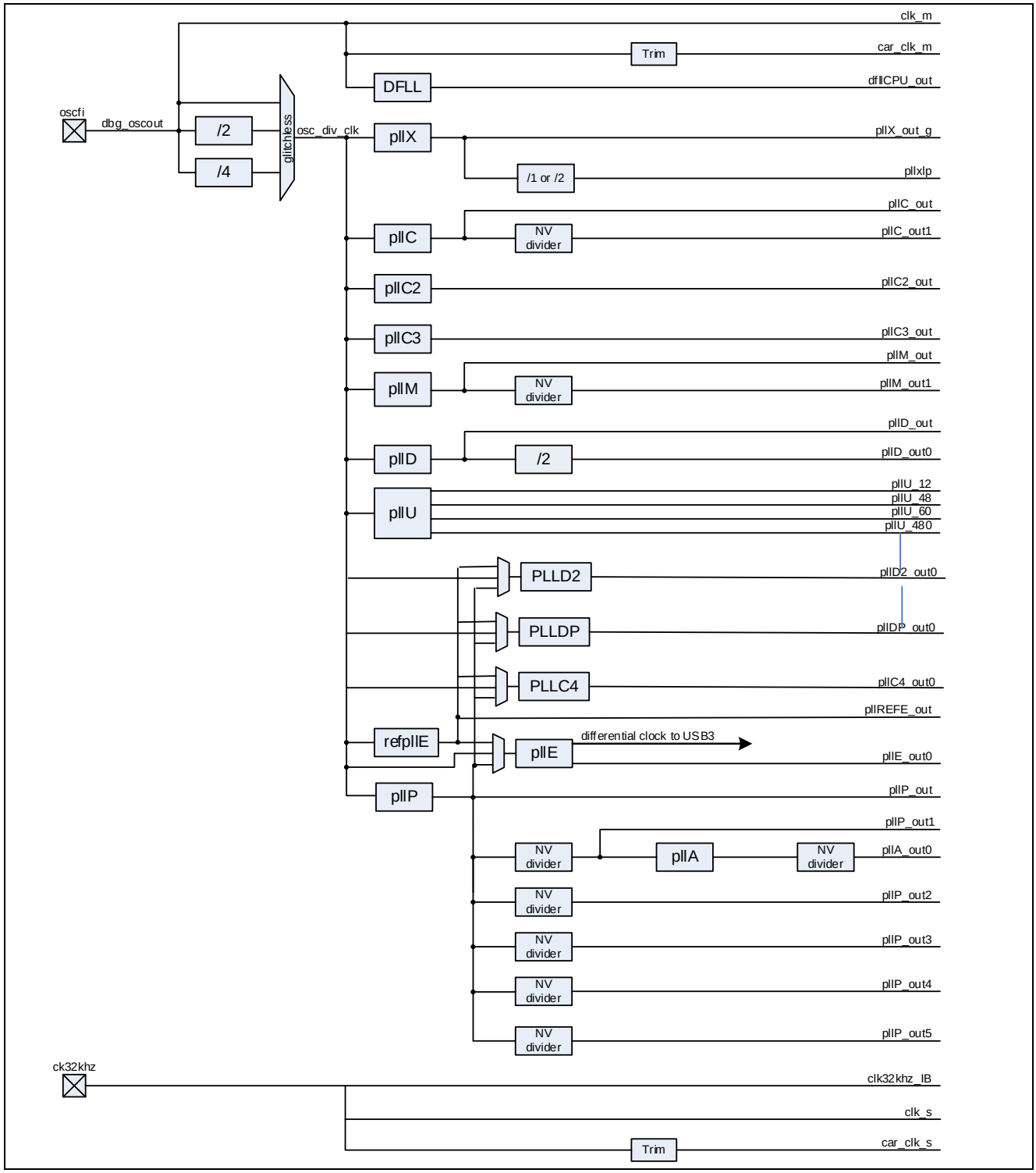
The table below lists which clock source they are derived from and their divider size.

Table 16: Clock Sources for the Derived Clocks

	dbg_oscout	pllC_out	ck32khz_lB	pllM_out	pllM_2x_out	pllP_out	int_pllA_out	Have super clock divider ? (# of bits)	Have U7.1 divider? (# of bits)
pllC_out1		x							8
pllM_out1				x					8
pllP_out1						x			8
pllP_out2						x			8
pllP_out3						x			8
pllP_out4						x			8
pllP_out5						x			8
pllA_out0							x		8

These clock sources are shown in the following figure.

Figure 4: Clocking Sources



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

Notes:

- Clock source pllX_out is an output of a mux that can select between the pllX_out clock and a predivide-by-2 pllX_out clock. Clock source dfllCPU_out is an output of a mux that can select between the dfllCPU_out clock and a predivide-by-2 dfllCPU_out.
- When EMC clock source = 4, the U7.1 divider is bypassed/ignored. This setting will give a very short low jitter clock path from pllM_out to EMC clock.
- When clock source = 14 or 15, in addition to being selected into the glitch-less switching logic in CAR, the selected clock will also bypass the U7.1 divider creating a short low jitter clock path to the output clock.
- The Clock Skipper is controlled by SOC Therm hardware.

Table 17: Root Clock Sources (1 of 2)

	oscout	pllC_out	pllC2_out	pllC3_out	ck32khz_IB	pllM_out	pllM_out_for_emc	pllP_out	pllP_out4_t	pllP_out3_t	pllX_out	dfllCPU_out	pllC_out1	pllP_out2	pllM_out1	pllA_out0	spdif_audio_2x_sync_clk	pllD_out0	pllD2_out0	pllE_clockout
actmon_clk_t	6	2	1	3	4			0												
adx0_r_clk	6	2	1	3				4								0				
adx1_r_clk	6	2	1	3				4								0				
amx0_r_clk	6	2	1	3				4								0				
amx1_r_clk	6	2	1	3				4								0				
audio_r_clk	6	2	1	3				4								0				
cilab_clk_t	6	2						0												
cilcd_clk_t	6	2						0												
cile_clk_t	6	2						0												
clk72mhz_clk	3	1	2							0										
csite_clk_t	6	2	1	3		4		0												
dam0_r_clk	6	2	1	3				4								0				
dam1_r_clk	6	2	1	3				4								0				
dam2_r_clk	6	2	1	3				4								0				
display_clk_t	6	4				1		0								3		2	5	
displayb_clk_t	6	4				1		0								3		2	5	
dsia_lp_clk_t	6	2						0												
dsib_lp_clk_t	6	2						0												
dvfs_ref_r_clk	6	2	1	3		4		0												

	oscout	pllC_out	pllC2_out	pllC3_out	ck32khz_IB	pllM_out	pllM_out for_emc	pllP_out	pllP_out4_t	pllP_out3_t	pllX_out	dfllCPU_out	pllC_out1	pllP_out2	pllM_out1	pllA_out0	spdif_audio_2x_sync_clk	pllD_out0	pllD2_out0	pllE_clockout
dvfs_soc_r_clk	6	2	1	3		4		0												
emc_dll_clk_t	3	1	5	6			4	2												
emc_latency_clk_t	3	1	5	6			4	2												
entropy_r_clk	1				2															3
extperiph1_clk	3				1			2								0				4
extperiph2_clk	3				1			2								0				4
extperiph3_clk	3				1			2								0				4
hda_r_clk	6	2	1	3		4		0												
hdmi_audio_clk_t	3	1	2																	
hdmi_clk_t	6	4				1		0								3		2	5	
host1x_clk_t		2	1	3		0		4								6				
hsi_clk_t	6	2	1	3		4		0												
i2c1_r_clk	6	2	1	3		4		0												
i2c2_r_clk	6	2	1	3		4		0												
i2c3_r_clk	6	2	1	3		4		0												
i2c4_r_clk	6	2	1	3		4		0												
i2c5_r_clk	6	2	1	3		4		0												
i2c6_r_clk	6	2	1	3		4		0												
i2c_slow_clk	6	2	1	3	4			0												
i2s0_r_clk	6							4								0				
i2s1_r_clk	6							4								0				
i2s2_r_clk	6							4								0				
i2s3_r_clk	6							4								0				
i2s4_r_clk	6							4								0				
int_emc_clk	3	1	5	6		0	4	2												
int_hda2codec_2x_clk	6	2	1	3		4		0												
isp_r_clk_t	6	1	4	5		0		2								3				
la_clk_t	6	2	1	3		4		0												
lvds0_pad_clockin_t	6	4				1		0								3		2	5	

	oscout	pllC_out	pllC2_out	pllC3_out	ck32khz_IB	pllM_out	pllM_out_for_emc	pllP_out	pllP_out4_t	pllP_out3_t	pllX_out	dfllCPU_out	pllC_out1	pllP_out2	pllM_out1	pllA_out0	spdif_audio_2x_sync_clk	pllD_out0	pllD2_out0	pllE_clockout
mselect_clk_t	6	2	1	3	5	4		0												
msenc_clk_t		2	1	3		0		4								6				
nor_r_clk	6	2	1	3		4		0												
owr_r_clk	6	2	1	3		4		0												
pex_txclkref																				
pex_txclkref_grp0																				
pex_txclkref_grp1																				
pex_txclkref_grp2																				
pex_txclkref_tms																				
pwm_r_clk	6	2	1	3	4			0												
sata_oob_clk_t	6	2				4		0												
sclk_sel	0	5			6				2				1	4	7					
sdmmc1_r_clk_t	6	2	1	3		4		0												5
sdmmc2_r_clk_t	6	2	1	3		4		0												5
sdmmc3_r_clk_t	6	2	1	3		4		0												5
sdmmc4_r_clk_t	6	2	1	3		4		0												5
se_clk_t	6	2	1	3		4		0								5				
soc_therm_t		1	4	5		0		2								3				
spdif_in_r_clk		2	1	3		4		0												
spdif_out_r_clk	6							4								0	2			
spi1_clk_t	6	2	1	3		4		0												
spi2_clk_t	6	2	1	3		4		0												
spi3_clk_t	6	2	1	3		4		0												
spi4_clk_t	6	2	1	3		4		0												
spi5_clk_t	6	2	1	3		4		0												
spi6_clk_t	6	2	1	3		4		0												
sys2hsio_sata_r_clk	6	2				4		0												
traceclkin_clk_t	6	2	1	3		4		0												
tsec_clk_t	6	2	1	3		4		0								5				

	oscout	pllC_out	pllC2_out	pllC3_out	ck32khz_IB	pllM_out	pllM_out for_emc	pllP_out	pllP_out4_t	pllP_out3_t	pllX_out	dfllCPU_out	pllC_out1	pllP_out2	pllM_out1	pllA_out0	spdif_audio_2x_sync_clk	pllD_out0	pllD2_out0	pllE_clockout
tsensor_r_clk	4	2	1	3	6			0												
uarta_r_clk	6	2	1	3		4		0												
uartb_r_clk	6	2	1	3		4		0												
uartc_r_clk	6	2	1	3		4		0												
uartd_r_clk	6	2	1	3		4		0												
vde_clk_t	6	2	1	3		4		0												
vfir_clk_t	6	2	1	3		4		0												
vi_clk_t		2	1	3		0		4								6				
vi_sensor2_clk		2	1	3		0		4								6				
vi_sensor_clk		2	1	3		0		4								6				
vic_clk_t	6	1	4	5		0		2								3				
xusb_120m_clk	7	4	5	6	2															
xusb_core_clk		3	2	4				1												
xusb_core_dev_clk		3	2	4				1												
xusb_falcon_clk		3	2	4				1												
xusb_fs_clk								4												

Table 18: Root Clock Sources (2 of 2)

	pllREFE_clockout	uhsic_clk480pll	fo_60m_out	fo_48m_out	audio_clk_src1111	dam0_clk_src1111	dam1_clk_src1111	dam2_clk_src1111	i2s0_audio_sync_clk	i2s1_audio_sync_clk	i2s2_audio_sync_clk	i2s3_audio_sync_clk	i2s4_audio_sync_clk	pex_pad_txclkref	pex_pad_txclkref_div1	pex_pad_txclkref_div2	pllC4_clockout	pllP_out0	sys2hsio_clk_m_sys_clk	Divide
actmon_clk_t																				8
adx0_r_clk																				8
adx1_r_clk																				8
amx0_r_clk																				8

	pllREFE_clockout	uhsic_clk480pll	fo_60m_out	fo_48m_out	audio_clk_src1111	dam0_clk_src1111	dam1_clk_src1111	dam2_clk_src1111	i2s0_audio_sync_clk	i2s1_audio_sync_clk	i2s2_audio_sync_clk	i2s3_audio_sync_clk	i2s4_audio_sync_clk	pex_pad_txclkref	pex_pad_txclkref_div1	pex_pad_txclkref_div2	pllC4_clockout	pllP_out0	sys2hsio_clk_m_sys_clk	Divide
amx1_r_clk																				8
audio_r_clk					7															8
cilab_clk_t																				8
cilcd_clk_t																				8
cile_clk_t																				8
clk72mhz_clk																				8
csite_clk_t																				8
dam0_r_clk						7														8
dam1_r_clk							7													8
dam2_r_clk								7												8
display_clk_t																				0
displayb_clk_t																				0
dsia_lp_clk_t																				8
dsib_lp_clk_t																				8
dvfs_ref_r_clk																				8
dvfs_soc_r_clk																				8
emc_dll_clk_t																				8
emc_latency_clk_t																				8
entropy_r_clk																		0		8
extperiph1_clk																				8
extperiph2_clk																				8
extperiph3_clk																				8
hda_r_clk																				8
hdmi_audio_clk_t																		0		8
hdmi_clk_t																				8
host1x_clk_t																				8
hsi_clk_t																				8
i2c1_r_clk																				16
i2c2_r_clk																				16

	pllREFE_clockout	uhsic_clk480pll	fo_60m_out	fo_48m_out	audio_clk_src1111	dam0_clk_src1111	dam1_clk_src1111	dam2_clk_src1111	i2s0_audio_sync_clk	i2s1_audio_sync_clk	i2s2_audio_sync_clk	i2s3_audio_sync_clk	i2s4_audio_sync_clk	pex_pad_txclkref	pex_pad_txclkref_div1	pex_pad_txclkref_div2	pllC4_clockout	pllP_out0	sys2hsio_clk_m_sys_clk	Divide
i2c3_r_clk																				16
i2c4_r_clk																				16
i2c5_r_clk																				16
i2c6_r_clk																				16
i2c_slow_clk																				8
i2s0_r_clk									2											8
i2s1_r_clk										2										8
i2s2_r_clk											2									8
i2s3_r_clk												2								8
i2s4_r_clk													2							8
int_emc_clk																				0
int_hda2codec_2x_clk																				8
isp_r_clk_t																	7			8
la_clk_t																				8
lvds0_pad_clockin_t																				8
mselect_clk_t																				8
msenc_clk_t																				8
nor_r_clk																				8
owr_r_clk																				8
pex_txclkref														0						8
pex_txclkref_grp0															1	0				8
pex_txclkref_grp1															1	0				8
pex_txclkref_grp2															1	0				8
pex_txclkref_tms															1	0				8
pwm_r_clk																				8
sata_oob_clk_t																				8
sclk_sel																		3		8
sdmmc1_r_clk_t																				8
sdmmc2_r_clk_t																				8

	pllREFE_clockout	uhsic_clk480pll	fo_60m_out	fo_48m_out	audio_clk_src1111	dam0_clk_src1111	dam1_clk_src1111	dam2_clk_src1111	i2s0_audio_sync_clk	i2s1_audio_sync_clk	i2s2_audio_sync_clk	i2s3_audio_sync_clk	i2s4_audio_sync_clk	pex_pad_txclkref	pex_pad_txclkref_div1	pex_pad_txclkref_div2	pllC4_clockout	pllP_out0	sys2hsio_clk_m_sys_clk	Divide
sdmmc3_r_clk_t																				8
sdmmc4_r_clk_t																				8
se_clk_t																				8
soc_therm_t																				8
spdif_in_r_clk																				8
spdif_out_r_clk																				8
spi1_clk_t																				8
spi2_clk_t																				8
spi3_clk_t																				8
spi4_clk_t																				8
spi5_clk_t																				8
spi6_clk_t																				8
sys2hsio_sata_r_clk																				8
traceclkin_clk_t																				8
tsec_clk_t																				8
tsensor_r_clk																				8
uarta_r_clk																				17
uartb_r_clk																				17
uartc_r_clk																				17
uartd_r_clk																				17
vde_clk_t																				8
vfir_clk_t																				8
vi_clk_t																	7			8
vi_sensor2_clk																				8
vi_sensor_clk																				8
vic_clk_t																				8
xusb_120m_clk	1	3																	0	8
xusb_core_clk	5																		0	8
xusb_core_dev_clk	5																		0	8

	pllREFE_clockout	uhsic_clk480pll	fo_60m_out	fo_48m_out	audio_clk_src1111	dam0_clk_src1111	dam1_clk_src1111	dam2_clk_src1111	i2s0_audio_sync_clk	i2s1_audio_sync_clk	i2s2_audio_sync_clk	i2s3_audio_sync_clk	i2s4_audio_sync_clk	pex_pad_txclkref	pex_pad_txclkref_div1	pex_pad_txclkref_div2	pllC4_clockout	pllP_out0	sys2hsio_clk_m_sys_clk	Divide
xusb_falcon_clk	5																		0	8
xusb_fs_clk		6		2															0	8

5.2.3 Generic Clock Core Structure

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

Changing the fractional divider ratio is also glitch-less and will use one of two mechanisms. When used in a switch with a clock source selection mux, the switching state-machine described above is used. When used without a clock source selection mux such as in the pllP_out1,2,3,4,5 dividers, the divider is not stopped and seamlessly switches to the new frequency at the end of a clock period. In this latter scheme, no dead cycles exist– between the original frequency and then the newly requested frequency.

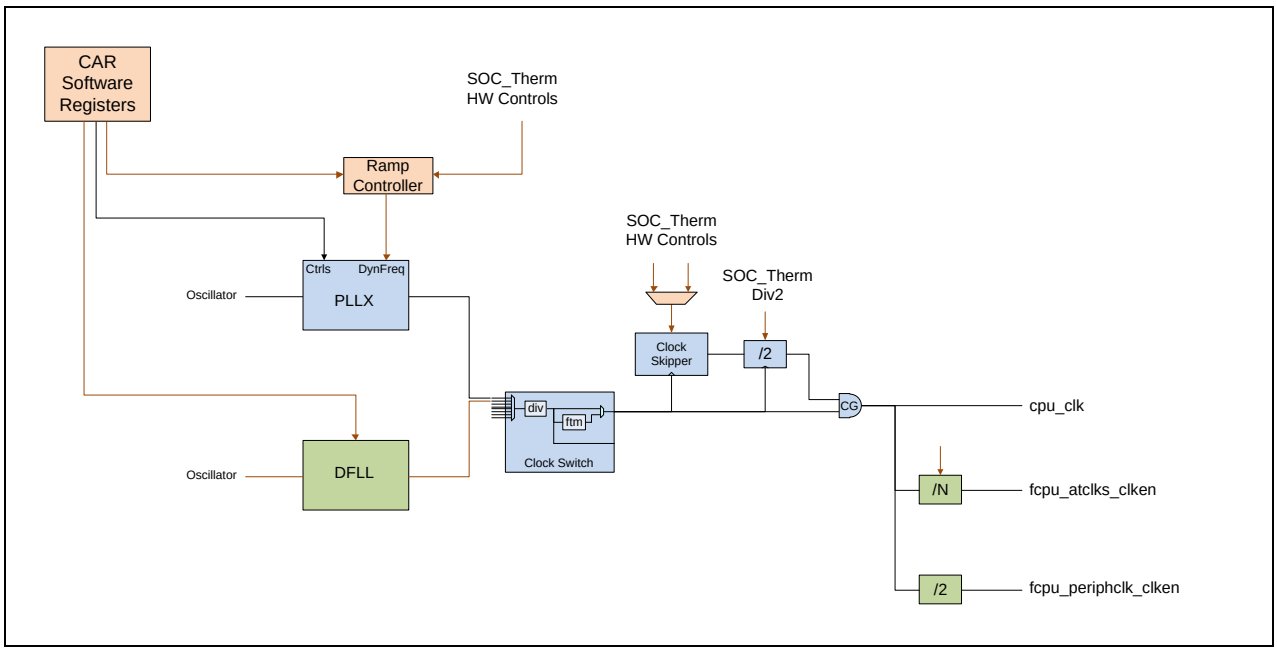
In all structures above, when the clock is gated off, it occurs at the end of the clock period so that 1) there are no runt pulses, and 2) the output clock will be stopped low.

5.2.4 Custom Core Clocks

5.2.4.1 FCPU Clock

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

Figure 5: FCPU Clock Generation



PLLX

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

DFLL

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

Clock Skipper

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

SOC Therm Div 2

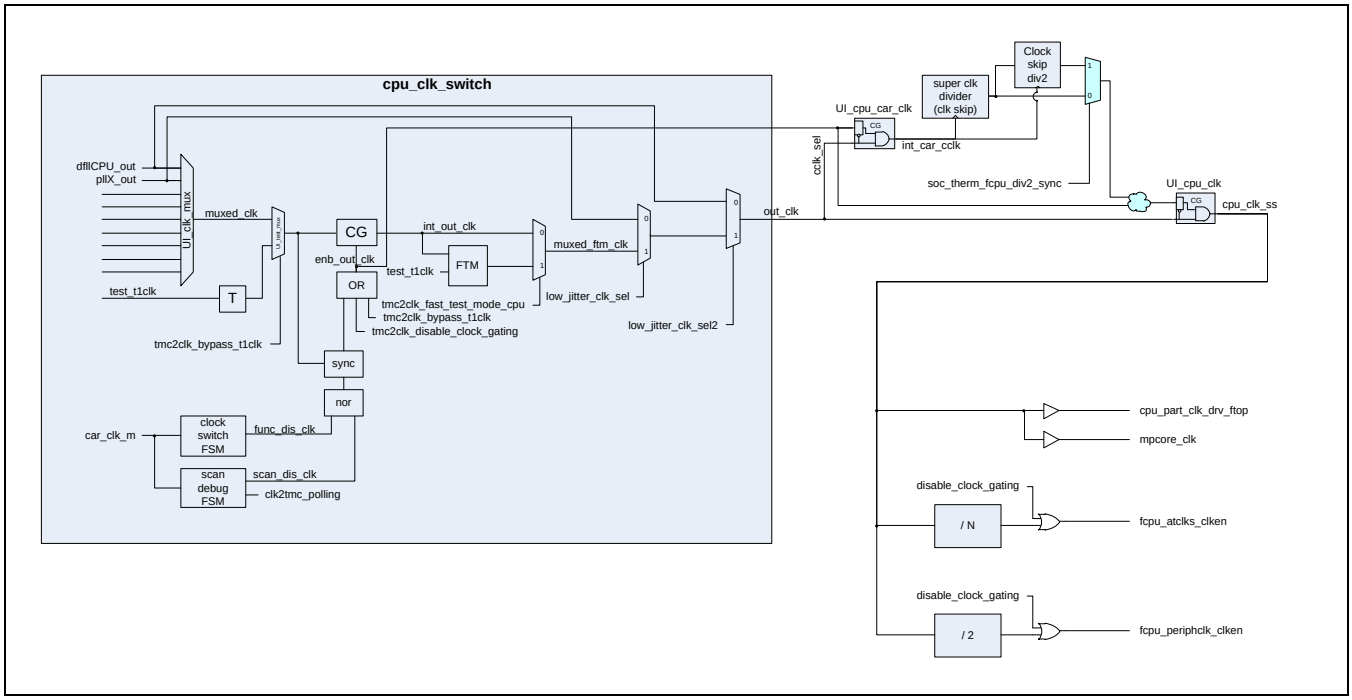
This logic will slow the CPU clock to manage thermal runaway.

Clock Switch

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

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

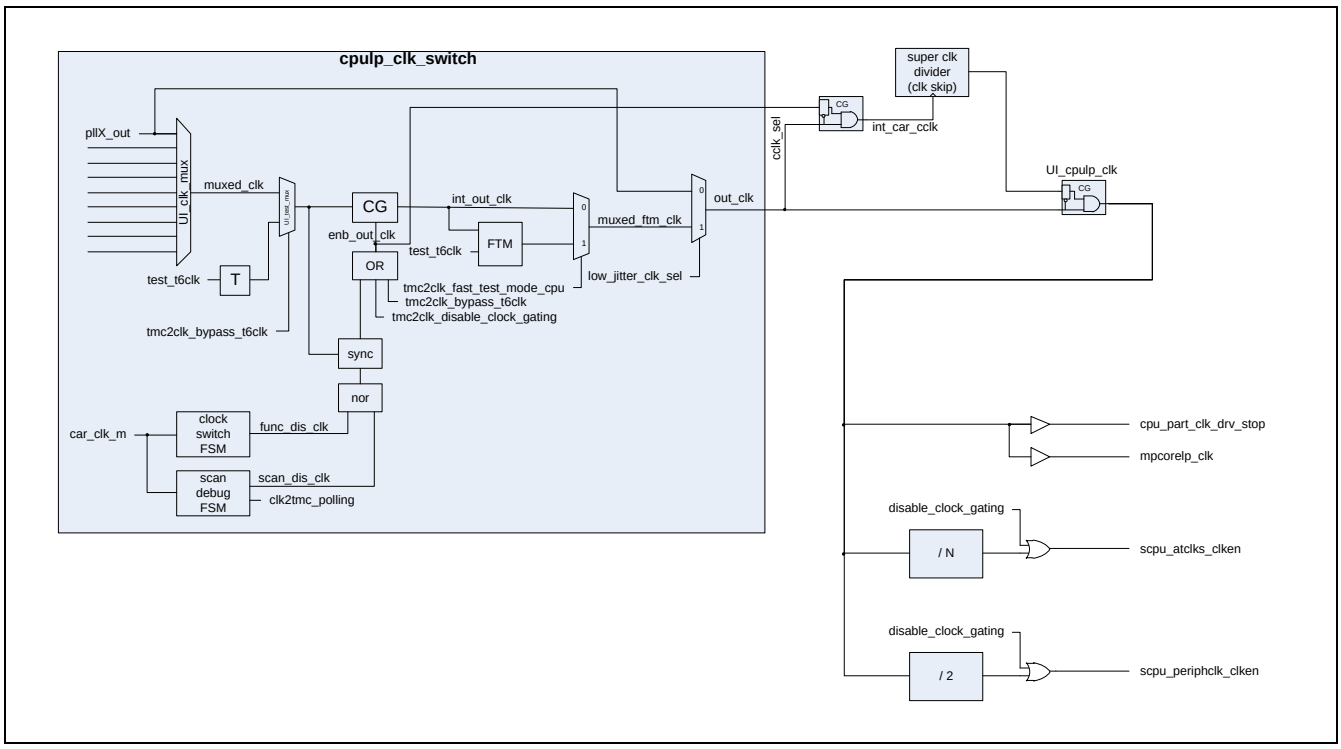
Figure 6: FCPU Clock Details



5.2.4.2 SCPU Clock

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

Figure 7: SCPU Clock Details



5.2.4.3 EMC/MC Clock

The PLLM's primary purpose is to clock the MC and EMC. The EMC/MC clocking structure is complicated due to the need to support high speed DRAMs, avoid clock switching when DRAMs are being accessed, and conserve power.

When changing the EMC frequency, internal logic sequencing handles the requirement that the DRAMs must be placed into self-refresh before the clock frequency is changed. Certain register fields when written do not get immediately applied but rather are applied at the correct time during the sequencing. These registers are referred to as "shadowed". Other register fields initiate the sequencing state-machine and therefore should be written last. At the end of the sequencing, the DRAMs are restored to being operational.

The reprogramming sequence is:

1. Program the MC/EMC shadow registers corresponding to the new frequency. These registers are "shadowed", as noted in their descriptions, and will not impact the hardware until later during the clock change sequence.
2. Program the EMC clock change FIFO (CCFIFO) with the pre-/post-clock change sequence.
3. Program CAR CLK_SOURCE_EMC register to trigger a clock change. At this point hardware takes the following actions:
   - The CAR block asserts the clock change request to the EMC.
   - The EMC stops MC transactions, flushes its internal outstanding requests, executes MRSs/enter self-refresh, and then updates the timing to copy the EMC shadow
   - Register contents (including CDB phase select) to EMC current register.
   - The EMC asserts clock change acknowledge to the CAR block.
   - The CAR stops the clock to PLLM/CDB, updates to the new clock source, and passes the EMC CDB phase select control to the CDB.
   - The CAR re-enables the clock to PLLM/CDB.
   - The CAR de-asserts the clock change request to the EMC (telling it the clock change is done).

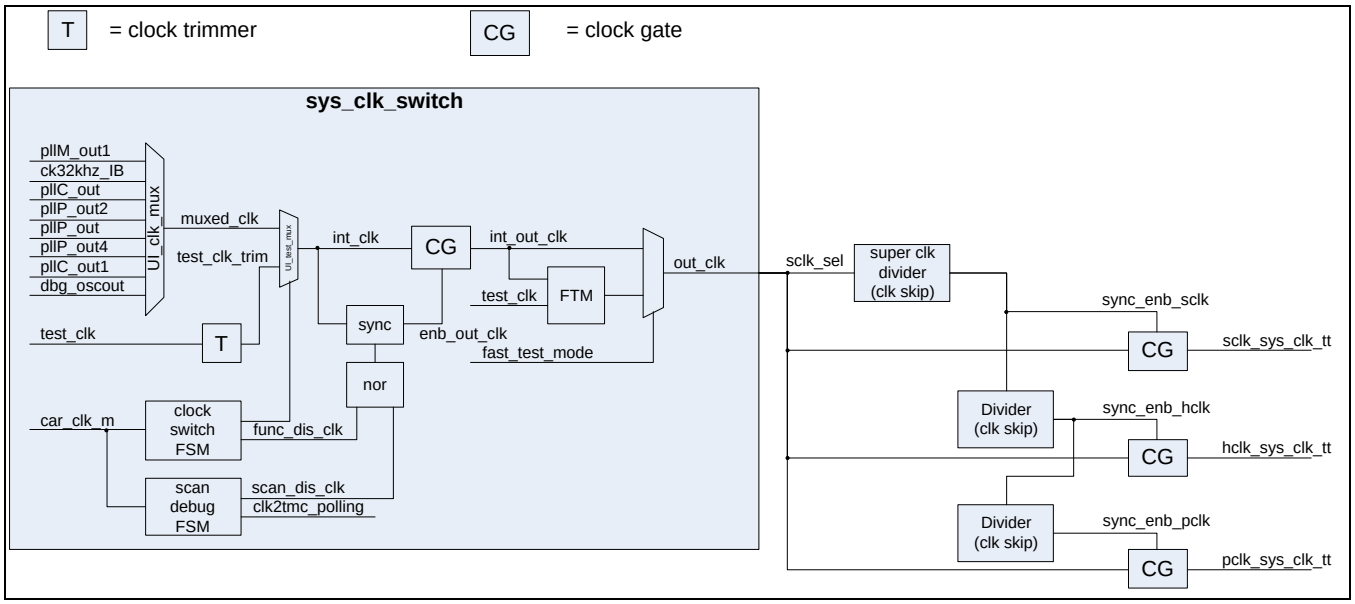
- The EMC unblocks MC transactions, de-asserts clock change acknowledge.
- The EMC executes the post clock change sequence (self-refresh exit, MRSs).
- Normal memory access resumes.

4. Software monitors the clock change complete interrupt status from the EMC register to know when the clock change is done.

5.2.4.4 SYS Clock Generation (SCLK, HCLK, PCLK)

The system clock sclk is used by the AVP sub-system (COP), hclk is the AHB bus clock, and pclk is the APB bus clock.

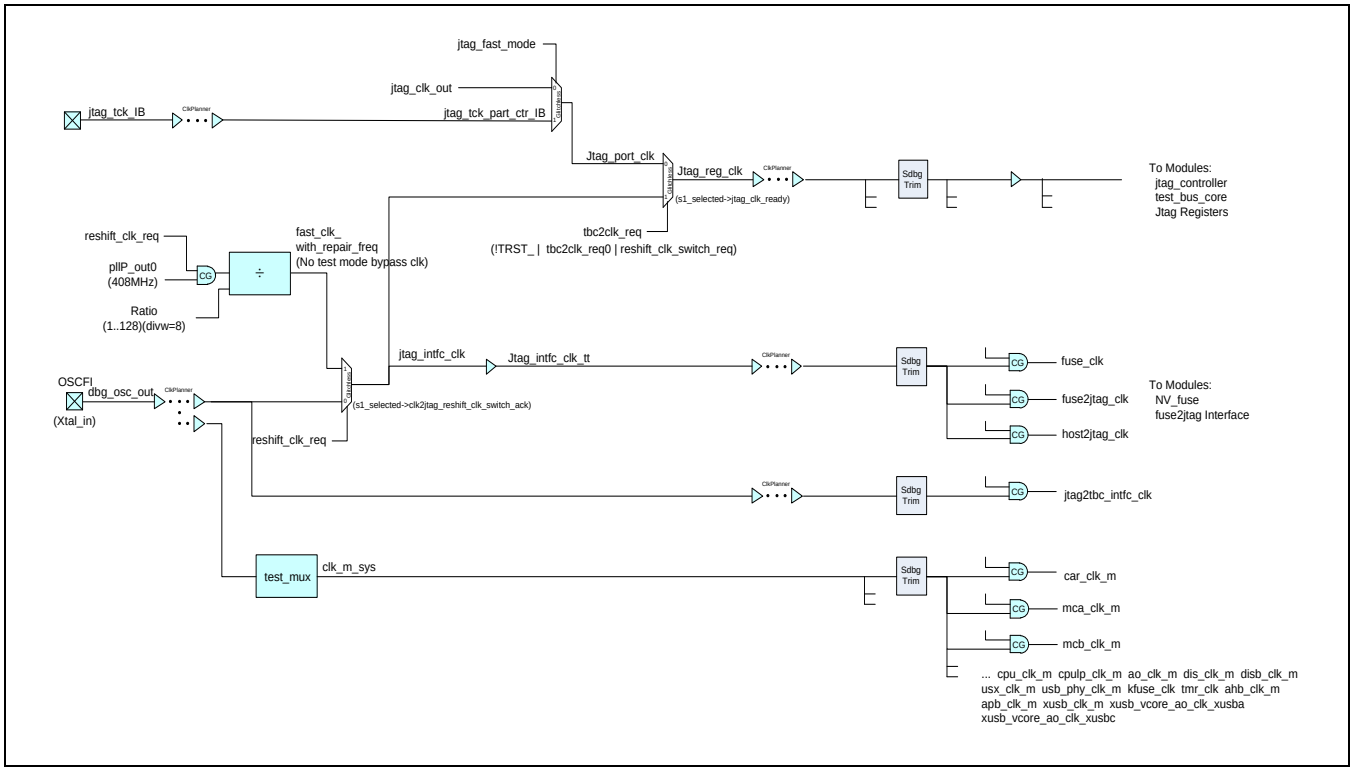
Figure 8: SCLK Clock Generation



5.2.4.5 Jtag_ref_clk and fuse_clk

The RAMs in the Fast CPU need to support RAM repair. Since the power rail to the FCPU will be shut down in modes such as LP1, the RAM repair information will be lost. When power is returned, the RAM repair information needs to be reprogrammed. A faster clock is needed to avoid delays in bringing the FCPU online. Additionally, a synchronous clock relationship is temporarily needed between the fuse and the re-repair logic running on the JTAG clock.

Figure 9: Jtag_ref_clk and fuse_clk Clock Generation



5.3 PLLs

5.3.1 DFLLCPU

DFLLCPU is a dedicated clock source for the Fast CPU. The DFLL is based on a ring oscillator and translates voltage changes into frequency compensation changes needed to prevent the CPU from failing.

5.3.2 PLLX

Tegra K1 devices require an alternative clock source in addition to DFLLCPU. Additionally, for EDP management and to avoid creating di/dt problems, this PLL needs a dynamic frequency changing mechanism.

Similar to prior Mobile products, PLLX is dedicated for this purpose. This PLL will feed to both the Fast CPU Cluster and Shadow CPU. This PLL cannot be easily used opportunistically for other units.

5.3.2.1 PLLX Startup Sequence

PLLX has additional logic for support of dynamic ramp. To improve leakage power, additional logic in the PLL is also power-gated with help of an IDDQ control. The sequence between IDDQ and ENABLE is detailed below.

1. After powering up, the PLL rails (in case of LP0 exit), change IDDQ from 1 to 0.

2. Wait for 2 µs.

3. Program PLL parameters in registers.

4. Change ENABLE from 0->1 while the reference clock is running and stable.

5. ENABLE=1 and IDDQ=1->0 is not allowed for starting PLL.

5.3.3 refPLLE, PLLE, USB3 Brick PLLs

USB 3.0 requires a spread spectrum (SS) clock. It also has stringent jitter requirements to be met. A spread-spectrum PLL with high VCO frequency (for low jitter) meets these requirements. A dedicated PLL (PLLE) is used for this purpose.

In addition to PLLE, to ensure that oscillator clock jitter gets filtered out prior to feeding it to PLLE, a reference PLL (refPLLE) is required. There are additional PLL(s) in the pad brick used for physical layer signaling for USB 3.0. For cases where spread spectrum might not be required and jitter may not be an issue, an option is kept to bypass PLLE to save power. This option requires that the crystal be 12 MHz or 48 MHz to be able to create the required PLLE output frequency of 100 MHz.

5.3.4 PLLU and UTMI PLL

USB 2.0 mode requires 8 phase data sampling logic per USB 2.0 port to capture data. This requires a dedicated PLL (UTMI_PLL) with 5 data sampling logic ports. XUSB needs 2 samplers (that is, one for host mode and one for device mode). Synopsys IP needs 3.

The dedicated PLL (PLLU) is used to generate the low jitter reference clock for UTMI_PLL. There is a provision to bypass PLLU for cases where a slightly higher jitter is acceptable for UTMI PLL reference clock. USB2 mode requires 8 phase data sampling logic per USB2 port to capture data. This requires a dedicated PLL (UTMI_PLL) with 5 data sampling logic ports.

5.3.5 PLLM

The DDR interface in Tegra K1 devices is required to operate at 1866 MT/s. This means that the EMC should work at 933 MHz and the MC should work at 466 MHz. The memory subsystem (DDR, EMC, and MC) is required to support a number of discrete frequencies such as DDR1866, DDR 1600, DDR 1333, DDR 1066, DDR 800 as well as slower frequencies to conserve power. A dedicated PLL (PLLM) is used for memory clocks.

DDR interface signaling also needs a DLL to support ¼ clock shift to reliably capture/drive DQ. In addition, a CDB (clock distribution buffer) is required to get a low jitter/skew clock for each byte/command group.

5.3.6 PLLA and PLLP

All audio clocks are a multiple of the sampling frequency which can be 32 KHz, 44.1 KHz or 48 KHz. A frequency of 56.448 MHz or 73.728 MHz can be used to generate any of these sampling rates. It is not possible to generate both of these frequencies from the osc_freq by just using PLLA so a cascade of 2 PLLs, PLLP to PLLA, is required to generate the audio clocks.

PLLP is fixed at generating 408 MHz, and PLLA takes in 9.6 MHz (PLLP / 42.5) to generate the two required audio frequencies.

Because PLLP generates a fixed frequency clock for audio, it can also be divided down and used for other modules which might need a fixed frequency clock.

Most audio codecs have their own PLL, which turns out to be more power efficient, so it is possible that PLLA does not get used in a typical system for audio. In such a case PLLA can be used as a general purpose PLL.

5.3.7 Display Clocks and PLLs (PLLD, PLLD2, and PLLDP)

The display requires three PLLs:

- PLLD (MIPI PLL) – Used to feed CSI/DSI pads.
- PLLD2 (SS PLL) – Used for HDMI
- PLLDP – Used for eDP/LVDS

PLLD is a dedicated PLL (with differential clock outputs and some DSI specific functionality) which should be used for DSI0 and DSI1. This PLL can be configured to generate the pixel clock from any oscillator frequency. These PLLs consume high

power (estimated at 4 times a regular general purpose PLL) and should be used only if DSI is required. For HDMI, the pad brick has PLLs which are used for interface signaling. These PLLs expect a pixel clock as an input.

5.3.7.1 PLLD and PLLD2 Startup Sequence

Additional logic in the PLL is also power-gated with help of an IDDQ control. The sequence between IDDQ and ENABLE is detailed below.

1. After powering up the PLL rails (in case of LP0 exit), change IDDQ from 1 to 0.
2. Wait for 2 µs.
3. Program PLL parameters in registers
4. Change ENABLE from 0->1 while the reference clock is running and stable.
5. ENABLE=1 and IDDQ=1->0 is not allowed

5.3.8 PLLC, PLLC2, PLLC3, and PLLC4

The AVP (300MHz), MSENC (333 MHz), imaging (VI/ISP – 600 MHz), VDE (366 MHz) and various other high frequency peripherals have different maximum frequency requirements. PLLC2 and PLLC3, can go up to 700 MHz, are used for these modules.

PLLC4 is used to provide 600 MHz dedicated output to VI and ISP blocks.

5.3.8.1 PLLC and PLLC4 Startup Sequence

PLLC has additional logic for support of dynamic ramp. Additional logic in the PLL is also power-gated with help of an IDDQ control. The sequence between IDDQ and ENABLE is detailed below.

1. After powering up the PLL rails (in case of LP0 exit), change IDDQ from 1 to 0.
2. Wait for 2 µs.
3. Program PLL parameters in registers.
4. Change ENABLE from 0->1 while reference clock is running and stable.
5. ENABLE=1 and IDDQ=1->0 is not allowed.

5.3.8.2 Digital PLLs (PLLC2, PLLC3) Startup Sequence

To save more power, additional power gating is implemented in digital PLLs; the additional ENABLE control has a corresponding register bit added.

1. PLL rails are turned ON (in case they are OFF when coming out from LP0)
2. IDDQ changes from 1 to 0 with ENABLE being 0
3. Program the PLL parameters
4. ENABLE is also asserted with guarantee that the reference clock is running.
5. ENABLE=1 and IDDQ=1->0 is not allowed.

5.3.9 GPU PLL

The GPU uses a dedicated GPCPLL inside the GPU core (PLL20G_DYN_PRB_ESD). This PLL has dynamic frequency scaling capabilities and generates a 400 MHz gpcclk.

5.4 Reset Architecture

The external PMC provides the primary power-on reset (SYS_RESET_N_). Apart from POR, some other events can also result in a system reset. These are:

1. Reset due to an indication from a thermal sensor. The thermal sensor module would assert a reset signal tsensor2pmc_reset, which would in turn generate a reset for the whole chip and in the process de-assert tsensor2pmc_reset itself.
2. Expiry of watchdog timer. The watchdog timer has a counter which is loaded with an initial value and ticks on periodic intervals. Once the counter expires, it reloads itself with the initial value, increments an expiry count, and generates an event for software. If software acknowledges the event, the expiry count is cleared and the counter is loaded back with initial value. This process continues indefinitely. Depending on the current expiry count, the event generated for software could be different. There are 2 types of watchdog timers in Tegra K1 systems depending on how they generate the reset:
   - Deadman timer (referred as WDT in reset code): This is the legacy implementation in which:

     1st expiry an interrupt is issued

     2nd expiry a reset is issued. This reset, however, resets only a subset of units.
   - Watchdog timer (referred as WDT2 in reset code):

     1st expiry – interrupt is issued

     2nd expiry – FIQ is issued

     3rd expiry – CPU reset is issued

     4th expiry – full system reset – This is the relevant reset for this section
3. Software reset. This reset is controlled by a configuration bit in the PMC address space (main_swrst). Once asserted, this result in a reset generation for the whole chip and in the process de-asserts itself.
4. LP0 wakeup reset. This is controlled by the logic within the PMC.

The POR (SYS_RESET_N_) is de-asserted by the external PMC after the power sequencing is done and after the RTC clock (clk_32KHz) and crystal clock is already running. For the other sources, the reset is originated within the Tegra K1 hardware.

5.5 Power Gating and Ungating

5.5.1 Clamp/Reset/Clock/PG-Enable Sequencing

5.5.1.1 Power Gating

For power-gating (powering off) a partition, the clamp, reset, clocks, and PG-enable (also known as sleep-enable) controls must be sequenced as shown in following diagram. For CCPLEX PG partitions, this sequencing is ensured by hardware (when power-gating is done via the flow controller). For SoC (non-CCPLEX) PG partitions, this sequencing needs to be ensured by software.

The following registers are used for the Clamp/PG-Enable control of each PG partition:

- PMC_PWRGATE_TOGGLE
- PMC_REMOVE_CLAMPING_CMD

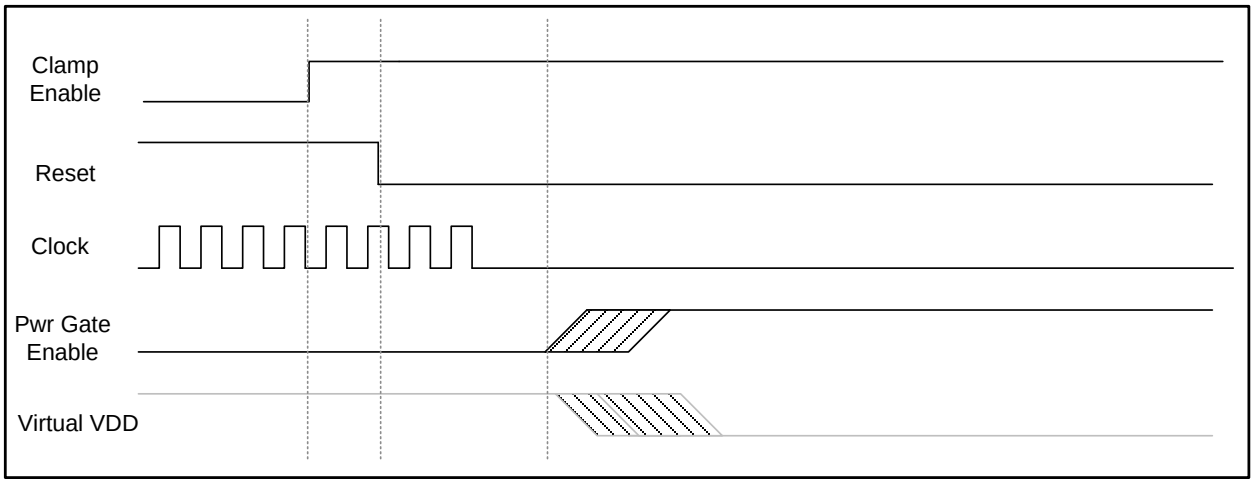
The following registers are used for the Clock/Reset control of each unit:

- RST_DEVICES_L/H/U/V/W/X
- CLK_OUT_ENB_L/H/U/V/W/X

In general, clamp-enable should be asserted before reset (as shown in the diagram). This works with synchronous or asynchronous clamping. However, if a unit can guarantee that its output signals will have the same clamp values as their pre-

clamp values (idle values) as well as their reset values, then clamp and reset ordering is not critical. Note it is non-trivial to verify the above guarantee.

Figure 10: Power-Gating Timing Sequence for Clamp/Reset/Clock/PG-Enable

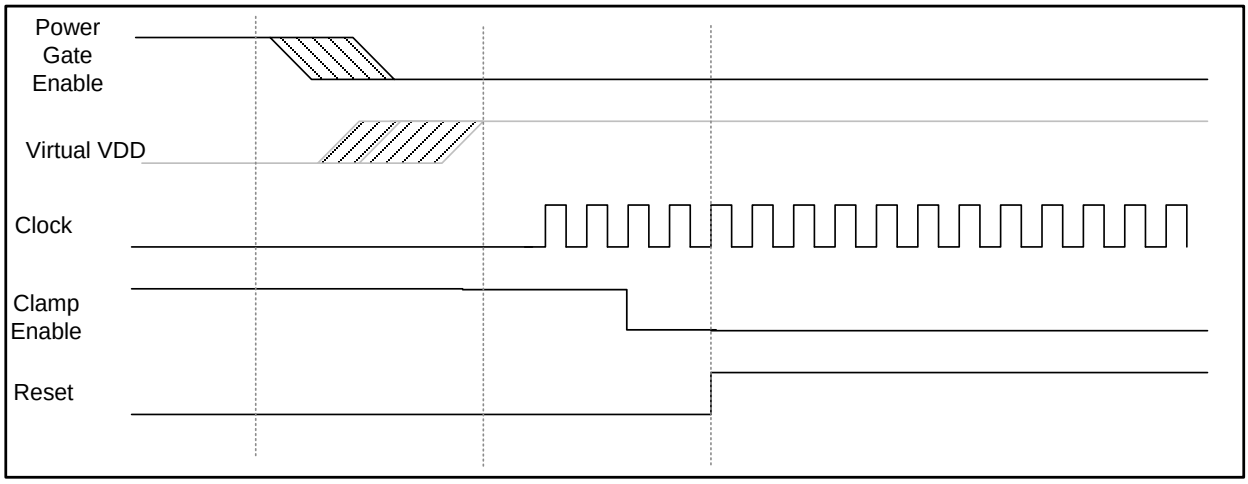


5.5.1.2 Power Ungating

For power-ungating (powering on) a partition, we have to ensure that clamp, reset, clocks, and PG-enable (sleep-enable) controls are sequenced as shown in the following diagram. For CCPLEX PG partitions, this sequencing is ensured by hardware (when power-gating is done via the flow controller). For SoC (non-CCPLEX) PG partitions, this sequencing needs to be ensured by software.

In general, the clamp enable should be de-asserted before reset (as shown in the diagram). This works with synchronous or asynchronous clamping. However, if a unit can guarantee that its output signals will have same clamp values as its post-clamp values (idle values), then clamp and reset ordering is not critical for power ungating. Note it is non-trivial to verify the above guarantee.

Figure 11: Power-Ungating Timing Sequence for Clamp/Reset/Clock/PG-Enable



5.5.2 Power-Gating Controller

The PMC provides power-gating controllers as programmable sequencers. When the power gating/ungating in is triggered, the controller sequences PG-enable signals with a programmable delay (measured in APB clock cycles) between consecutive

"zones" PG-enable controls. There are two types of power-gating controllers: CPU and SOC power-gating controllers. The following sections describe both types of controllers.

5.5.2.1 SOC Power-Gating Controller

The SOC power-gating controller is used to sequence power gating of SOC (including shadow CPU cluster) power partitions. It uses 8 zones and shares the same programming register for inter-zone delays. The zones are powered up in one order and powered down in reverse order, using the same inter-zone timings. So only one set of delays are used for both powering down and up.

For all non-CPU SOC partitions, power-gating can only be turned on/off by direct register write. For CPU (shadow) SOC partitions, power-gating can be turned on/off via flow-controller. Refer to "SOC Power Gating Controllers" in the PMC section.

GPU Power Gating

The GPU power gating is controlled by the GPMU unit inside the Kepler GPU. This is independent of the SOC/CPU power-gating control.

If the GPU has its own rail, then the software mutual exclusion for GPU ELPG and SoC power gating is not needed. But for a heavily cost reduced system where the GPU and SoC might share a rail, the software must interlock the two mechanisms in order to avoid di/dt problems (which could result if the two mechanisms were to alter power gating state simultaneously).

5.5.2.2 Fast CPU Power-Gating Controller

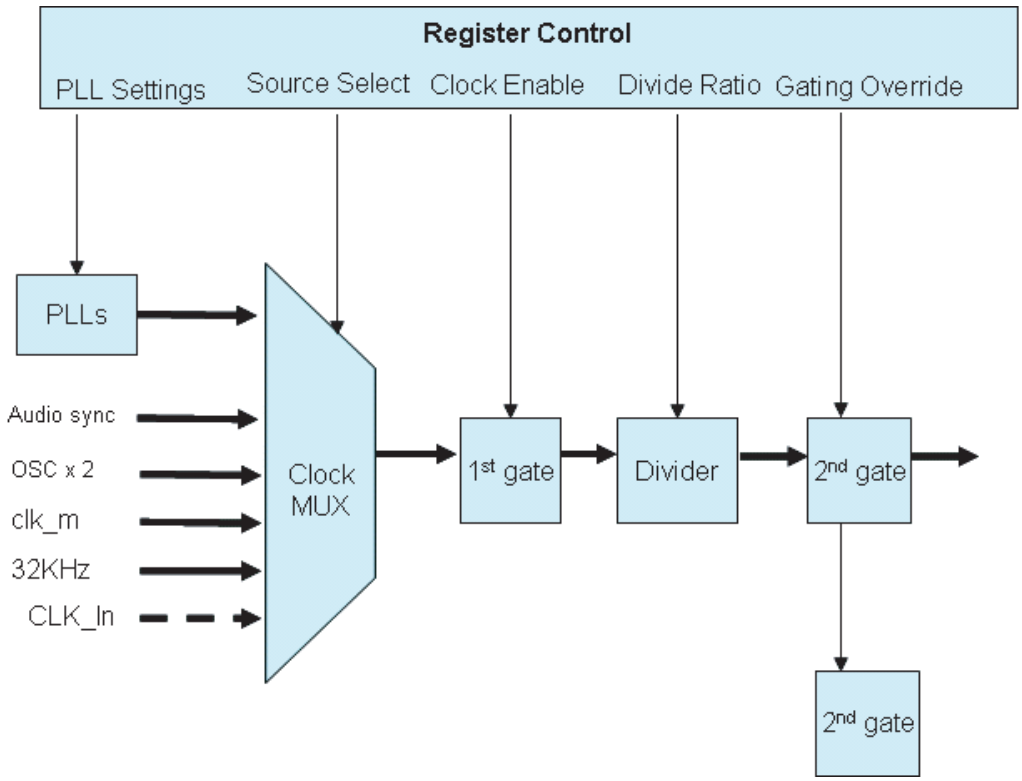
The fast CPU power-gating controller is used to sequence power-gating of fast CPU partitions. All CPU partitions (CE0/1/2/3 and C0NC) share the same programming register for inter-zone delays, which are different from inter-zone delays in SoC partitions. The zones are powered up in one order and powered down in reverse order, using the same inter-zone timings. So only one set of delays are used for both powering down and up. For fast CPU partitions, power gating can be turned on/off via the flow controller. Refer to "CPU-G Power Gating Controllers" in the PMC section.

5.6 Software Features and Programming Model

5.6.1 Clock Control

The figure below illustrates the software clock control model.

Figure 12: Clock Control



5.6.2 PLL Programming

PLL output frequencies are programmed by setting their N, M, and P values. The governing equations are:

VCO = (Fi / M) * N,

Fo = VCO / (2 ^ P)

where *Fo* is the output frequency from the PLL.

> **Note:** Not all PLLs have the simple mapping for the P value listed above. Contact your NVIDIA representative for assistance.

There are three requirements for each PLL that must be complied with:

- Each PLL has a legal input frequency (*Fi*) range.
- Each PLL has a legal comparison frequency (*CF*) range, where *CF = Fi / M*
- Each PLL has a legal *VCO* frequency range, where *VCO = CF * N*.

To change a PLL, do the following:

- Ensure that no enabled module is using the PLL that will change.
- Program the new PLL settings
- Wait for PLL stabilization
- Change the divider values for each clock that will use the PLL to divide-down to the target frequency, and change the module clock sources for all modules that will use the new PLL settings.

5.6.3 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)

5.6.4 Changing Clock Sources and Clock Dividers

Changing a clock source (except the clock sources to the audio_sync_clk) to a running module is "glitch free". The clock source and divide ratio can be set concurrently without creating spurious frequencies. For the audio_sync_clk, the clock source select needs to be set up before changing the device clock source to use that audio clock or to enable the device which uses that audio clock.

The glitch-free clock divider to a running module can also be changed. All modules support changing the clock divider ratio without disabling the clock; write the new divider to the appropriate register See the section on Power below.

To change the clock source and divider (either after system reset or for any other reason), follow this sequence:

Sequence 1 to change clock source and divider:

1. Assert reset to the module if it is not already asserted
2. Enable clock to the module if the clock is not already enabled
3. Change the clock divider and/or the clock source to the module
4. Wait 2 µs for the clock to flush through the pipe / logic.
5. De-assert reset to the module.

5.6.5 Clock Gating (Override)

Please refer to the register definition section for more information.

5.6.5.1 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.

5.6.6 Programming Guide for Power Gating and Ungating

This section covers the software procedures for power gating and ungating of SOC power domains. Power gating and ungating of CPU power domains is not covered here because it is handled through the Flow Controller.

5.6.6.1 Procedure Summary

The general procedure for power gating an SOC power domain is as follows:

1. Write MC register to flush and block MC clients
2. Write CAR register to assert unit resets
3. Write CAR register to disable clocks
4. Write PMC register to gate power domain

The general procedure for power ungating an SOC power domain is as follows:

1. Write PMC register to ungate power domain
2. Write CAR register to enable clocks
3. Write CAR register to de-assert resets
4. Write MC register to enable MC clients

In some cases, partitions require specific deviations from the general procedure. Deviations are covered in the domain-specific sections below.

This table summarizes the respective clock and reset bits and MC clients in each SOC power domain.

Table 19: MC Clients for Clocks and Resets per Domain

Domain	Units	MC Clients
VE	VI, CSI, ISP	VI, ISP
VE2	ISPB	ISP2B
VDE	VDE	VDE
PCX	AFI	AFI
MPEA	MSENC	MSENC
SAX	SATA	SAX

Domain	Units	MC Clients
DIS	DISPLAY	DC
DISB	DISPLAYB	DCB
SOR	MIPI_CAL, DPAUX, SOR, HDMI, DSI, DSIB	None
XUSBB	XUSB_DEV	XUSB_DEV
XUSBC	XUSB_HOST	XUSB_HOST
VIC	VIC	VIC

5.6.6.2 Procedures for VE Power Domain

VE Power Gating

1. Flush MC clients VI and ISP by setting the following bits:
   - MC_CLIENT_HOTRESET_CTRL_0.VI_FLUSH_ENABLE
   - MC_CLIENT_HOTRESET_CTRL_0.ISP2_FLUSH_ENABLE
   - MC_CLIENT_HOTRESET_CTRL_1_0.ISP2B_FLUSH_ENABLE

   Also Poll MC_CLIENT_HOTRESET_STATUS_1_0 until ISP2B_HOTRESET_STATUS is set.
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bits are set:
   - VI_HOTRESET_STATUS
   - ISP2_HOTRESET_STATUS
3. Set the following bits to assert reset to VI, ISP, ISPB, and CSI:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_VI
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_ISP
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_X_0.CLK_ENB_ISPB
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_CSI
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_CILAB
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_CILCD
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_CILE
4. Clear the following bits to disable clocks to VI, ISP, ISPB, and CSI:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_VI
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_ISP
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_CSI
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_CILAB
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_CILCD
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_CILE
5. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = VE
   - START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the VE bit is set

VE Power Ungating

1. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = VE

- START = ENABLE

2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the VE bit is cleared
3. Set the following bits to enable clocks to VI, ISP, ISPB, and CSI:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_VI
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_ISP
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_X_0.CLK_ENB_ISPB
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_CSI
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_CILAB
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_CILCD
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_CILE
4. Remove power-gating clamps by writing a 1 to the following bit:
   - APBDEV_PMC_REMOVE_CLAMPING_CMD_0.VE
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the VE bit is cleared
6. If needed, clear the following bits to de-assert the reset to VI, ISP, ISPB, and CSI:
   - CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_VI_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_ISP_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_X_0.SWR_ISPB_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_CSI_RST
7. Enable MC clients VI and ISP by clearing the following bits:
   - MC_CLIENT_HOTRESET_CTRL_0.VI_FLUSH_ENABLE
   - MC_CLIENT_HOTRESET_CTRL_0.ISP2_FLUSH_ENABLE
   - MC_CLIENT_HOTRESET_CTRL_1_0.ISP2B_FLUSH_ENABLE

5.6.6.3 Procedures for VDE Power Domain

VDE Power Gating

1. Flush the VDE MC client by setting the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.VDE_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bit is set:
   - VDE_HOTRESET_STATUS
3. Set the following bits to assert the reset to the VDE:
   - CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_VDE_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_BSEV_RST
4. Clear the following bits to disable clocks to the VDE:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_VDE
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_BSEV
5. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = VDE
   - START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the VDE bit is set

VDE Power Ungating

1. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = VDE
   - START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the VDE bit is cleared
3. Set the following bits to enable clocks to the VDE:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_VDE
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_BSEV
4. Remove power-gating clamps by writing a 1 to the following bit:
   - APBDEV_PMC_REMOVE_CLAMPING_CMD_0.VDE
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the VDE bit is cleared
6. Clear the following bits to de-assert the reset to the VDE:
   - CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_VDE_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_BSEV_RST
7. Enable the VDE MC client by clearing the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.VDE_FLUSH_ENABLE

5.6.6.4 Procedures for PCX Power Domain

PCX Power Gating

1. Flush the AFI MC client by setting the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.AFI_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bit are set:
   - AFI_HOTRESET_STATUS
3. Set the following bits to assert the reset to the PCIE and AFI:
   - CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_PCIE_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_AFI_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_PCIEXCLK_RST
4. Clear the following bits to disable clocks to PCIE and AFI:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_PCIE
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_PCIERX0
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_PCIERX1
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_PCIERX2
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_PCIERX3
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_PCIERX4
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_AFI
5. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = PCX
   - START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the PCX bit is set

PCX Power Ungating

1. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = PCX
   - START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the PCX bit is cleared
3. Set the following bits to enable clocks to PCIE:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_PCIE
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_PCIERX0
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_PCIERX1
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_PCIERX2
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_PCIERX3
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_PCIERX4
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_AFI
4. Remove power-gating clamps by writing a 1 to the following bit:
   - APBDEV_PMC_REMOVE_CLAMPING_CMD_0.PCX
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the PCX bit is cleared
6. Clear the following bits to de-assert the reset to the PCX:
   - CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_PCIE_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_AFI_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_PCIEXCLK_RST
7. Enable the AFI MC client by clearing the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.AFI_FLUSH_ENABLE

5.6.6.5 Procedures for MPE Power Domain

MPE Power Gating

1. Flush the MSENC MC client by setting the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.MSENC_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bit is set:
   - MSENC_HOTRESET_STATUS
3. Set the following bit to assert the reset to the MSENC:
   - CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_MSENC_RST
4. Clear the following bit to disable clocks to the MSENC:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_MSENC
5. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = MPE
   - START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the MPE bit is set

MPE Power Ungating

1. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = MPE

- START = ENABLE

2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the MPE bit is cleared
3. Set the following bits to enable clocks to the MSENC:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_MSENC
4. Remove power-gating clamps by writing a 1 to the following bit:
   - APBDEV_PMC_REMOVE_CLAMPING_CMD_0.MPE
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the MPE bit is cleared
6. Clear the following bits to de-assert the reset to the MSENC:
   - CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_MSENC_RST
7. Enable the MSENC MC client by clearing the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.MSENC_FLUSH_ENABLE

5.6.6.6 Procedures for SAX Power Domain

SAX Power Gating

1. Flush the SATA MC clients by setting the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.SATA_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bit is set:
   - SATA_HOTRESET_STATUS
3. Set the following bits to assert the reset to SATA:
   - CLK_RST_CONTROLLER_RST_DEVICES_V_0.SWR_SATA_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_W_0.SWR_SATACOLD_RST
4. Clear the following bit to disable clocks to SATA:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_V_0.CLK_ENB_SATA
5. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = SAX
   - START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the SAX bit is set

SAX Power Ungating

1. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = SAX
   - START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the SAX bit is cleared
3. Set the following bit to enable clocks to SATA:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_V_0.CLK_ENB_SATA
4. Remove power-gating clamps by writing a 1 to the following bit:
   - APBDEV_PMC_REMOVE_CLAMPING_CMD_0.SAX
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the SAX bit is cleared
6. Clear the following bits to de-assert the reset to SAX:
   - CLK_RST_CONTROLLER_RST_DEVICES_V_0.SWR_SATA_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_W_0.SWR_SATACOLD_RST

7. Enable the SATA MC clients by clearing the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.SATA_FLUSH_ENABLE

5.6.6.7 Procedures for DIS Power Domain

DIS Power Gating

The VE power domain has to be gated before the DIS can be gated.

1. Flush the DC MC client by setting the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.DC_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bit is set:
   - DC_HOTRESET_STATUS
3. Set the following bit to assert the reset to DISP1:
   - CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_DISP1_RST
4. Clear the following bit to disable clocks to DISP1:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_DISP1
5. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = DIS
   - START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the DIS bit is set

DIS Power Ungating

DIS power ungating requires the SOR to be ungated as well.

1. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   - PARTID = DIS
   - START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the DIS bit is cleared
3. Set the following bit to enable clocks to DISP1:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_DISP1
4. Remove power-gating clamps by writing a 1 to the following bit:
   - APBDEV_PMC_REMOVE_CLAMPING_CMD_0.DIS
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the DIS bit is cleared
6. Clear the following bit to de-assert the reset to DISP1:
   - CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_DISP1_RST
7. Enable the DC MC client by clearing the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.DC_FLUSH_ENABLE

5.6.6.8 Procedures for DISB Power Domain

DISB Power Gating

1. Flush the DCB MC client by setting the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.DCB_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bit is set:

- DCB_HOTRESET_STATUS

3. Set the following bit to assert the reset to DISP2:
   - CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_DISP2_RST
4. Clear the following bits to disable clocks to DISP2:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_DISP2
5. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = DISB
   - START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the DISB bit is set

DISB Power Ungating

DISB power ungating requires the SOR to be ungated as well.

1. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   - PARTID = DISB
   - START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit DISB is clear
3. Set the following bits to enable clocks to DISP2:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_DISP2
4. Remove power-gating clamps by writing a 1 to the following bit:
   - APBDEV_PMC_REMOVE_CLAMPING_CMD_0.DISB
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the DISB bit is cleared
6. Clear the following bits to de-assert reset to DISP2:
   - CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_DISP2_RST
7. Enable the DCB MC client by clearing the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.DCB_FLUSH_ENABLE

5.6.6.9 Procedures for SOR Power Domain

SOR Power Gating

SOR can only be power gated when both DIS and DISB are gated.

1. Set the following bits to assert the reset to MIPI_CAL, DPAUX, SOR, HDMI, DP2, DSI and DSIB:
   - CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_MIPI_CAL_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_X_0.SWR_DPAUX_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_X_0.SWR_SOR0_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_HDMI_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_DSI_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_DSIB_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_W_0.SWR_DP2_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_W_0.SWR_ HDA2HDMICODEC_RST
2. Clear the following bits to disable clocks to MIPI_CAL, DPAUX, SOR, HDMI, DP2, DSI and DSIB:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_MIPI_CAL
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_X_0.CLK_ENB_DPAUX

- CLK_RST_CONTROLLER_CLK_OUT_ENB_X_0.CLK_ENB_SOR0
- CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_HDMI
- CLK_RST_CONTROLLER_CLK_OUT_ENB_X_0.CLK_ENB_HDMI_AUDIO
- CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_DSI
- CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_DSIB
- CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_DSIA_LP
- CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_DSIB_LP
- CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_DP2
- CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0. CLK_ENB_HDA2HDMICODEC

3. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = SOR
   - START = ENABLE
4. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the SOR bit is set

SOR Power Ungating

1. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = SOR
   - START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the SOR bit is cleared
3. Set the following bits to enable clocks to MIPI_CAL, DPAUX, SOR, HDMI, DP2, DSI and DSIB:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_MIPI_CAL
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_X_0.CLK_ENB_DPAUX
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_X_0.CLK_ENB_SOR0
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_HDMI
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_X_0.CLK_ENB_HDMI_AUDIO
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_DSI
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_DSIB
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_DSIA_LP
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_DSIB_LP
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_DP2
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_HDA2HDMICODEC
4. Remove power-gating clamps by writing a 1 to the following bit:
   - APBDEV_PMC_REMOVE_CLAMPING_CMD_0.SOR
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the SOR bit is cleared
6. Clear the following bits to de-assert the reset to MIPI_CAL, DPAUX, SOR, HDMI, DP2, DSI and DSIB:
   - CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_MIPI_CAL_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_X_0.SWR_DPAUX_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_X_0.SWR_SOR0_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_HDMI_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_DSI_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_DSIB_RST

- CLK_RST_CONTROLLER_RST_DEVICES_W_0.SWR_DP2_RST
- CLK_RST_CONTROLLER_RST_DEVICES_W_0.SWR_HDA2HDMICODEC_RST

5.6.6.10 Procedures for XUSBA Power Domain

XUSBA Power Gating

1. Set the following bit to assert the reset to XUSB_SS:
   - CLK_RST_CONTROLLER_RST_DEVICES_W_0.SWR_XUSB_SS_RST
2. Clear the following bit to disable clocks to XUSB_SS:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_XUSB_SS
3. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = XUSBA
   - START = ENABLE
4. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the XUSBA bit is set

XUSBA Power Ungating

1. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = XUSBA
   - START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the XUSBA bit is cleared
3. Set the following bit to enable clocks to XUSB_SS:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_XUSB_SS
4. Remove power-gating clamps by writing a 1 to the following bit:
   - APBDEV_PMC_REMOVE_CLAMPING_CMD_0.XUSBA
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the XUSBA bit is cleared
6. Clear the following bit to de-assert the reset to XUSB_SS:
   - CLK_RST_CONTROLLER_RST_DEVICES_W_0.SWR_XUSB_SS_RST

5.6.6.11 Procedures for XUSBB Power Domain

XUSBB Power Gating

Software should be aware that disabling XUSB_DEV clock will also affect the XUSB_HOST module that is in XUSBC power domain.

1. Flush XUSB_DEV MC client by setting the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.XUSB_DEV_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bit is set:
   - XUSB_DEV_HOTRESET_STATUS
3. Set the following bit to assert reset to XUSB_DEV:
   - CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_XUSB_DEV_RST
4. Clear the following bits to disable clocks to XUSB_DEV:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_XUSB_DEV
5. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = XUSBB

- START = ENABLE

6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the XUSBB bit is set

XUSBB Power Ungating

1. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = XUSBB
   - START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the XUSBB bit is cleared
3. Set the following bit to enable clocks to XUSB_DEV:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_XUSB_DEV
4. Remove power-gating clamps by writing a 1 to the following bit:
   - APBDEV_PMC_REMOVE_CLAMPING_CMD_0.XUSBB
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the XUSBB bit is cleared
6. Clear the following bit to de-assert the reset to XUSB_DEV:
   - CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_XUSB_DEV_RST
7. Enable the XUSB_DEV MC client by clearing the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.XUSB_DEV_FLUSH_ENABLE

5.6.6.12 Procedures for XUSBC Power Domain

XUSBC Power Gating

Software should be aware that disabling XUSB_HOST clock will also affect the XUSB_SS module that is in the XUSBA power domain.

1. Flush the XUSB_HOST MC client by setting the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.XUSB_HOST_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bit is set:
   - XUSB_HOST_HOTRESET_STATUS
3. Set the following bits to assert the reset to XUSB_HOST:
   - CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_XUSB_HOST_RST
4. Clear the following bits to disable clocks to XUSB_HOST:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_XUSB_HOST
5. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = XUSBC
   - START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the XUSBC bit is set

XUSBC Power Ungating

1. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   - PARTID = XUSBC
   - START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the XUSBC bit is cleared
3. Set the following bit to enable clocks to XUSB_HOST:

- CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_XUSB_HOST

4. Remove power-gating clamps by writing a 1 to the following bit:
   - APBDEV_PMC_REMOVE_CLAMPING_CMD_0.XUSBC
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the XUSBC bit is cleared
6. Clear the following bit to de-assert the reset to XUSB_HOST:
   - CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_XUSB_HOST_RST
7. Enable the XUSB_HOST MC client by clearing the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.XUSB_HOST_FLUSH_ENABLE

5.6.6.13 Procedures for VIC Power Domain

VIC Power Gating

1. Flush the VIC MC client by setting the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.VIC_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bit is set:
   - VIC_HOTRESET_STATUS
3. Set the following bit to assert the reset to the MSENC:
   - CLK_RST_CONTROLLER_RST_DEVICES_X_0.SWR_VIC_RST
4. Clear the following bit to disable clocks to the VIC:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_X_0.CLK_ENB_VIC
5. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = VIC
   - START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the VIC bit is set

VIC Power Ungating

1. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = VIC
   - START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the VIC bit is cleared
3. Set the following bit to enable clocks to the VIC:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_X_0.CLK_ENB_VIC
4. Remove power-gating clamps by writing a 1 to the following bit:
   - APBDEV_PMC_REMOVE_CLAMPING_CMD_0.VIC
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the VIC bit is cleared
6. Clear the following bits to de-assert the reset to the VIC:
   - CLK_RST_CONTROLLER_RST_DEVICES_X_0.SWR_VIC_RST
7. Enable the VIC MC client by clearing the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.VIC_FLUSH_ENABLE

5.6.6.14 Procedures for IRAM Power Domain

IRAM Power Gating

1. Set the following bit to enable first level clock gating when IRAMs are not in use:
   - CLK_RST_CONTROLLER_ CLK_OUT_ENB_U_0.CLK_ENB_IRAM[A|B|C|D]
2. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = IRAM
   - START = ENABLE
3. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the IRAM bit is set

IRAM Power Ungating

1. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = IRAM
   - START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the IRAM bit is cleared
3. Set the following bit to disable first level clock gating:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_ U_0.CLK_ENB_ IRAM[A|B|C|D]
4. Remove power-gating clamps by writing a 1 to the following bit:
   - APBDEV_PMC_REMOVE_CLAMPING_CMD_0.IRAM
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the IRAM bit is cleared

5.7 Clock and Reset Controller Registers

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

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

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

Similarly for clocks, the appropriate bits in CLK_OUT_ENB_L/H/U/V/W/X 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:

- The ARM7 AVP Processor (COP) and its L1 cache
- All system buses (PPSB, AHB, APB, etc.)
- Timer
- RTC
- NOR Flash controller
- eFUSE
- GPIO
- CoreSight

Each of the boot block devices will have their reset de-asserted 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 CPU/ARM7 for another 511 oscillator clock periods. This will prevent either processor from talking to a boot device while it is still in the reset state.

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

5.7.1 Precautions

- Unless noted elsewhere, all modules support changing the clock divider ratio without disabling the clock.
- Unless noted elsewhere, all modules' clock switching is glitch-free except "audio_sync_clk".
- For "audio_sync_clk", the clock source select needs to be set up before changing the device clock source to use that audio clock or to enable the device which use that audio clock.
- Before stopping the clock (via CLK_OUT_ENB_L/H/U/V/W/X registers) and/or asserting reset (via RST_DEVICES_L/H/U/V/W/X registers) to a module, first check the module to make sure it is not active. Stopping clock/asserting reset while the module is still busy can cause relatively minor problem such as incorrect data read/written, or catastrophic problem such as system hang. To ensure a module is not active, (a) disable the module by programming its disable bit if not already done so, and (b) wait until the module is not active by checking for its busy bit, done bit, count, or similar mechanism.

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

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

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

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

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

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

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

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

To reset a device (without need to change clock) after boot-up:

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

Method Used to wait for 1 or 2 µs.

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

Invert Duty-Cycle Distortion Control

Some dividers have a feature to help mitigate clock duty-cycle distortion -- the distortion of the positive edge of a clock can be swapped with the distortion of the negative edge. This ability may prove useful to counteract some of the inherent distortion that occurs in clock generation logic and distribution. For example, a divider with a ratio of 1.0 having a 51%/49% duty cycle could be made to have a 49%/51% duty cycle, if the duty-cycle inversion control is activated.

- **Frequency Change Caution**:

  An INVERT_DCD field can be changed while the clock is running. In circumstances described in item 3 below, this may temporarily cause a negative impact on the divided clock frequency, causing the module to fail if the frequency to the module is not temporarily adjusted first.

  1. Asserting INVERT_DCD adds 1/2 clock period (divider input clock) to the low pulse of a single divided output clock period during the change. This causes the divided output clock frequency to be lowered for one clock period.
  2. For ratios other than 1.0, de-asserting INVERT_DCD removes 1/2 clock period (divider input clock) from the low pulse of a single divided output clock period during the change. Because this effectively speeds up the divided output clock frequency for one clock period, which can cause a module to fail, the module clock frequency may need to be changed to a lower frequency before de-asserting INVERT_DCD.
  3. De-asserting INVERT_DCD with a ratio of 1.0 is unique in that removing 1/2 clock period (divider input clock) from the low pulse effectively removes the low pulse altogether. The resulting combination of the high pulses on either side will effectively look like 1/2 clock period was added to the high pulse instead. Thus there occurs a lower frequency for one clock period rather than an increased frequency like all other divide ratios.

- **Latency**:

  Once the INVERT_DCD signal gets to the divider logic, there will be a maximum delay of 6 divider input clocks + 1 divider output clock before the change will complete.

Main clock sources used by the system:

(A) Primary clocks.

- "osc" or "clk_m" which can be either 12 MHz, 13 MHz, 19.2 MHz 26 MHz, 16.8 MHz, 38.4 MHz, or 48 MHz (Not all the frequencies listed under osc or clk_m can be met).
- "clk_s" which is 32 kHz.

(B) PLL clocks.

- "PLLC" is general purpose and its output is called "pllC_out".
- "PLLC2" is general purpose and its output is called "pllC2_out".
- "PLLC3" is general purpose and its output is called "pllC3_out".
- "PLLC4" is general purpose and its output is called "pllC4_out".
- "PLLM" (memory) is primarily used for memory and its output is called "pllM_out".
- "PLLP" (peripheral) has a fixed frequency and its output is called "pllP_out0".
- "PLLA" (audio) is cascaded from PLLP and is used for audio purposes.
- "PLLU" (USB) has fixed outputs at 12 MHz, 48 MHz, 60 MHz, and 480 MHz.
- "PLLD"(DSI) is primarily used for the display and its output (after a /2 fix) is called "pllD_out0".
- "PLLD2"(DSI) is primarily used for the display and its output (after a /2 fix) is called "pllD2_out0".
- "PLLX" has extra high frequency used primarily by the fast CPU and is called "pllX_out0".
- “PLLDP” is used for eDP/LVDS (spread spectrum) and is called “pllDP_out0”.
- "refPLLE" is only used by PLLE.
- "PLLE" is only used by PCIe, SATA, USB3 (if they are present)

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

- "pllC_out1" is divided-down from "pllC_out0".
- "pllM_out1" is divided-down from "pllM_out0".
- "pllP_out1" is divided-down from "pllP_out0".
- "pllP_out2" is divided-down from "pllP_out0".
- "pllP_out3" is divided-down from "pllP_out0".
- "pllP_out4" is divided-down from "pllP_out0".
- "pllP_out5" is divided-down from "pllP_out0".
- "pllA_out0" is divided-down from the "pllA" output.

Note: With the exception of PLLA, the other PLLs all use "osc_div_clk" as the reference clock. This is the same frequency as "osc" with the exception of the following crystals:

38.4 MHz - Needs to be divided by 2 providing 19.2 MHz to the PLLs.

48.0 MHz - Needs to be divided by 4 providing 12.0 MHz to the PLLs

Note: In this document, CPULP, SCPU, and Slow CPU are synonymous. CPUG, FCPU, and Fast CPU are also synonymous.

5.7.2 Multi-Address Tagging

A register or bit marked with a [* Multi-Address] tag indicates multiple addresses can access the same register/bit. The two types of Multi-Address tags are described below.

[CCLK Multi-Address] Tags

These tags apply to legacy, CPUG, and CPULP multiple address registers and bits. For CCLK (CPU_CLK) related registers values:

- Three sets of addresses are provided to support single cluster legacy, CPUG, and CPULP clusters.
- Two sets of hardware flops are used to store programmed values for G and LP CPU clusters. Legacy single cluster registers are aliased to one of these based on the active cluster at the time the register is referenced.
- In some cases, only the CPU bit in an otherwise single access register is multi-address accessible, for example, CLK_ENB_CPU.

[FUSE Multi-Address] Tags

These tags apply to Fuse and CAR multiple address registers:

- A bond_out_* register can be programmed via fuse or CAR CLK_RST_CONTROLLER_BOND_OUT_L_0 register or fuse FUSE_SKU_BOND_OUT_L_0 register.
- Fuse bond_out writes are updated on fuse2all_fuse_outputs_valid rising edge
- CAR bond_out writes are updated on register writes

5.7.3 CLK_RST_CONTROLLER_RST_SOURCE_0

WatchDog (Deadman) Timer

The watchdog timer is used to recover from hang/lockup condition by resetting the system and/or COP (ARM7) and/or CPU (MPCore). Either timer1 or timer2 can be used as watchdog timer. Refer to the Timer section of this document for more information on watchdog timer usage.

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

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

Bit	R/W	Reset	Description
			0 = TIMER1 1 = TIMER2
2	RW	DISABLE	WDT_SYS_RST_EN: Enable WatchDog Timer reset for system. 0 = DISABLE 1 = ENABLE
1	RW	DISABLE	WDT_COP_RST_EN: Enable WatchDog Timer reset for COP 0 = DISABLE 1 = ENABLE
0	RW	DISABLE	WDT_CPU_RST_EN: Enable WatchDog Timer reset for CPU 0 = DISABLE 1 = ENABLE

5.7.4 CLK_RST_CONTROLLER_RST_DEVICES_L_0

Offset: 0x4 │ Read/Write: R/W │ Reset: 0x7cd7dXXX (0b011111xx11x1x11111x1111x11xx100x)

Bit	R/W	Reset	Description
31	RW	DISABLE	SWR_CACHE2_RST: Reset COP cache controller. 0 = DISABLE 1 = ENABLE
30	RW	ENABLE	SWR_I2S0_RST: Reset I2S0 Controller 0 = DISABLE 1 = ENABLE
29	RW	ENABLE	SWR_VCP_RST: Reset vector coprocessor. 0 = DISABLE 1 = ENABLE
28	RW	ENABLE	SWR_HOST1X_RST: Reset HOST1X. 0 = DISABLE 1 = ENABLE
27	RW	ENABLE	SWR_DISP1_RST: Reset DISP1 controller. 0 = DISABLE 1 = ENABLE
26	RW	ENABLE	SWR_DISP2_RST: Reset DISP2 controller. 0 = DISABLE 1 = ENABLE
23	RW	ENABLE	SWR_ISP_RST: Reset ISP controller. 0 = DISABLE 1 = ENABLE
22	RW	ENABLE	SWR_USBD_RST: Reset USB controller. 0 = DISABLE 1 = ENABLE
20	RW	ENABLE	SWR_VI_RST: Reset VI controller. 0 = DISABLE 1 = ENABLE
18	RW	ENABLE	SWR_I2S2_RST: Reset I2S 2 Controller 0 = DISABLE 1 = ENABLE
17	RW	ENABLE	SWR_PWM_RST: Reset Pulse Width Modulator 0 = DISABLE 1 = ENABLE
15	RW	ENABLE	SWR_SDMMC4_RST: Reset SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	RW	ENABLE	SWR_SDMMC1_RST: Reset SDMMC1 controller.

Bit	R/W	Reset	Description
			0 = DISABLE 1 = ENABLE
12	RW	ENABLE	SWR_I2C1_RST: Reset I2C1 Controller 0 = DISABLE 1 = ENABLE
11	RW	ENABLE	SWR_I2S1_RST: Reset I2S 1 Controller 0 = DISABLE 1 = ENABLE
10	RW	ENABLE	SWR_SPDIF_RST: Reset SPDIF Controller 0 = DISABLE 1 = ENABLE
9	RW	ENABLE	SWR_SDMMC2_RST: Reset SDMMC2 Controller 0 = DISABLE 1 = ENABLE
8	RO	X	SWR_GPIO_RST: Reset GPIO Controller 0 = DISABLE 1 = ENABLE
7	RW	ENABLE	SWR_UARTB_RST: Reset UARTB/VFIR Controller 0 = DISABLE 1 = ENABLE
6	RW	ENABLE	SWR_UARTA_RST: Reset UARTA Controller 0 = DISABLE 1 = ENABLE
5	RO	X	SWR_TMR_RST: Reset Timer Controller 0 = DISABLE 1 = ENABLE
3	RW	ENABLE	SWR_ISPB_RST: Reset ISPB controller
2	RW	DISABLE	SWR_TRIG_SYS_RST: Write 1 to pulse System Reset Signal. Hardware clears this bit. 0 = DISABLE 1 = ENABLE
1	RW	DISABLE	SWR_COP_RST: Write 1 to force COP Reset Signal. Software needs to clear this bit when done. 0 = DISABLE 1 = ENABLE
0	RO	X	SWR_CPU_RST: Tied to 0.

5.7.5 CLK_RST_CONTROLLER_RST_DEVICES_H_0

Offset: 0x8 │ Read/Write: R/W │ Reset: 0xefddf32X (0b111x111111x111x11111x0110x1xx11x)

Bit	R/W	Reset	Description
31	RW	ENABLE	SWR_BSEV_RST: Reset BSEV controller. 0 = DISABLE 1 = ENABLE
30	RW	ENABLE	SWR_BSEA_RST: Reset BSEA controller. 0 = DISABLE 1 = ENABLE
29	RW	ENABLE	SWR_VDE_RST: Reset VDE and BSEV controller. 0 = DISABLE 1 = ENABLE
27	RW	ENABLE	SWR_USB3_RST: Reset USB3 controller. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
26	RW	ENABLE	SWR_USB2_RST: Reset USB2 controller. 0 = DISABLE 1 = ENABLE
25	RW	ENABLE	SWR_EMC_RST: Reset EMC controller. 0 = DISABLE 1 = ENABLE
24	RW	ENABLE	SWR_MIPI_CAL_RST: Reset MIPI CAL Logic. 0 = DISABLE 1 = ENABLE
23	RW	ENABLE	SWR_UARTC_RST: Reset UARTC controller 0 = DISABLE 1 = ENABLE
22	RW	ENABLE	SWR_I2C2_RST: Reset I2C2 controller. 0 = DISABLE 1 = ENABLE
20	RW	ENABLE	SWR_CSI_RST: Reset CSI controller. 0 = DISABLE 1 = ENABLE
19	RW	ENABLE	SWR_HDMI_RST: Reset HDMI 0 = DISABLE 1 = ENABLE
18	RW	ENABLE	SWR_HSI_RST: Reset MIPI base-band HSI controller. 0 = DISABLE 1 = ENABLE
16	RW	ENABLE	SWR_DSI_RST: Reset DSI controller 0 = DISABLE 1 = ENABLE
15	RW	ENABLE	SWR_I2C5_RST: Reset I2C5 controller 0 = DISABLE 1 = ENABLE
14	RW	ENABLE	SWR_SPI3_RST: Reset SPI3 controller. 0 = DISABLE 1 = ENABLE
12	RW	ENABLE	SWR_SPI2_RST: Reset SPI2 controller. 0 = DISABLE 1 = ENABLE
10	RW	DISABLE	SWR_SNOR_RST: Reset NOR Flash controller. 0 = DISABLE 1 = ENABLE
9	RW	ENABLE	SWR_SPI1_RST: Reset SPI1 controller. 0 = DISABLE 1 = ENABLE
8	RW	ENABLE	SWR_KFUSE_RST: Reset KFuse controller. 0 = DISABLE 1 = ENABLE
5	RW	ENABLE	SWR_STAT_MON_RST: Reset statistic monitor. 0 = DISABLE 1 = ENABLE
2	RW	ENABLE	SWR_APBDMA_RST: Reset APB-DMA. 0 = DISABLE 1 = ENABLE
1	RW	ENABLE	SWR_AHBDMA_RST: Reset AHB-DMA. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
0	RO	X	SWR_MEM_RST: Reset MC. This bit is disabled for security reasons. You can write to it but it will always read as 0.

5.7.6 CLK_RST_CONTROLLER_RST_DEVICES_U_0

Offset: 0xc │ Read/Write: R/W │ Reset: 0x8a8ed5fe (0b1xxx1x1x1xxx111x110101011111111x)

Bit	Reset	Description
31	ENABLE	SWR_XUSB_DEV_RST: Reset XUSB DEV logic. 0 = DISABLE 1 = ENABLE
27	ENABLE	SWR_MSENC_RST: Reset MSENC logic. 0 = DISABLE 1 = ENABLE
25	ENABLE	SWR_XUSB_HOST_RST: Reset XUSB HOST logic. 0 = DISABLE 1 = ENABLE
23	ENABLE	SWR_EMUCIF_RST: Reset EMUCIF logic. 0 = DISABLE 1 = ENABLE
19	ENABLE	SWR_TSEC_RST: Reset TSEC logic. 0 = DISABLE 1 = ENABLE
18	ENABLE	SWR_DSIB_RST: Reset DSIB 0 = DISABLE 1 = ENABLE
17	ENABLE	SWR_I2C_SLOW_RST: Reset I2C_SLOW 0 = DISABLE 1 = ENABLE
15	ENABLE	SWR_DTV_RST: Reset DTV 0 = DISABLE 1 = ENABLE
14	ENABLE	SWR_SOC_THERM_RST: Reset SOC_THERM. 0 = DISABLE 1 = ENABLE
13	DISABLE	SWR_TRACECLKIN_RST: Reset CoreSight™ traceclkin controller. 0 = DISABLE 1 = ENABLE
11	DISABLE	SWR_AVPUCQ_RST: Reset AVPUCQ logic. 0 = DISABLE 1 = ENABLE
10	ENABLE	SWR_PCIEXCLK_RST: Reset PCIEXCLK logic. 0 = DISABLE 1 = ENABLE
9	DISABLE	SWR_CSITE_RST: Reset CoreSight controller. 0 = DISABLE 1 = ENABLE
8	ENABLE	SWR_AFI_RST: Reset AFI controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	ENABLE	SWR_OWR_RST: Reset OWR controller. 0 = DISABLE 1 = ENABLE
6	ENABLE	SWR_PCIE_RST: Reset PCIe® controller. 0 = DISABLE 1 = ENABLE
5	ENABLE	SWR_SDMMC3_RST: Reset SDMMC3 controller. 0 = DISABLE 1 = ENABLE
4	ENABLE	SWR_SPI4_RST: Reset SPI4 controller. 0 = DISABLE 1 = ENABLE
3	ENABLE	SWR_I2C3_RST: Reset I2C3 controller. 0 = DISABLE 1 = ENABLE
2	0x1	SWR_UARTE_RST: Reserved.
1	ENABLE	SWR_UARTD_RST: Reset UARTD controller. 0 = DISABLE 1 = ENABLE

5.7.7 CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x80000130 (0b100000xx00x0x00000x0000100110xx0)

Bit	Reset	Description
31	ENABLE	CLK_ENB_CACHE2: Enable clock to COP cache controller. 0 = DISABLE 1 = ENABLE
30	DISABLE	CLK_ENB_I2S0: Enable clock to I2S0 Controller 0 = DISABLE 1 = ENABLE
29	DISABLE	CLK_ENB_VCP: Enable clock to vector coprocessor. 0 = DISABLE 1 = ENABLE
28	DISABLE	CLK_ENB_HOST1X: Enable clock to Host1x. 0 = DISABLE 1 = ENABLE
27	DISABLE	CLK_ENB_DISP1: Enable clock to DISP1 controller. 0 = DISABLE 1 = ENABLE
26	DISABLE	CLK_ENB_DISP2: Enable clock to DISP2 controller. 0 = DISABLE 1 = ENABLE
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

Bit	Reset	Description
20	DISABLE	CLK_ENB_VI: Enable clock to VI controller. 0 = DISABLE 1 = ENABLE
18	DISABLE	CLK_ENB_I2S2: Enable clock to I2S2 controller 0 = DISABLE 1 = ENABLE
17	DISABLE	CLK_ENB_PWM: Enable clock to PWM (Pulse Width Modulator) 0 = DISABLE 1 = ENABLE
15	DISABLE	CLK_ENB_SDMMC4: Enable clock to SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	DISABLE	CLK_ENB_SDMMC1: Enable clock to SDMMC1 controller. 0 = DISABLE 1 = ENABLE
12	DISABLE	CLK_ENB_I2C1: Enable clock to I2C1 Controller 0 = DISABLE 1 = ENABLE
11	DISABLE	CLK_ENB_I2S1: Enable clock to I2S1 Controller 0 = DISABLE 1 = ENABLE
10	DISABLE	CLK_ENB_SPDIF: Enable clock to S/PDIF Controller 0 = DISABLE 1 = ENABLE
9	DISABLE	CLK_ENB_SDMMC2: Enable clock to SDMMC2 controller. 0 = DISABLE 1 = ENABLE
8	ENABLE	CLK_ENB_GPIO: Enable clock to GPIO Controller 0 = DISABLE 1 = ENABLE
7	DISABLE	CLK_ENB_UARTB: Enable clock to UARTB/VFIR Controller 0 = DISABLE 1 = ENABLE
6	DISABLE	CLK_ENB_UARTA: Enable clock to UARTA Controller 0 = DISABLE 1 = ENABLE
5	ENABLE	CLK_ENB_TMR: Enable clock to Timer Controller (sticky) 0 = DISABLE 1 = ENABLE
4	ENABLE	CLK_ENB_RTC: Enable clock to RTC Controller 0 = DISABLE 1 = ENABLE
3	DISABLE	CLK_ENB_ISPB: Enable clock – ISPB 0 = DISABLE 1 = ENABLE
0	DISABLE	CLK_ENB_CPU: Enable clock to CPU. [CCLK Multi-Address] 0 = DISABLE 1 = ENABLE

5.7.8 CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000480 (0b000x000000x000x00000x1001000x000)

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
27	DISABLE	CLK_ENB_USB3: Enable clock to USB3 controller. 0 = DISABLE 1 = ENABLE
26	DISABLE	CLK_ENB_USB2: Enable clock to USB2 controller. 0 = DISABLE 1 = ENABLE
25	DISABLE	CLK_ENB_EMC: Enable clock to MC/EMC controller. 0 = DISABLE 1 = ENABLE
24	DISABLE	CLK_ENB_MIPI_CAL: Enable clock to MIPI CAL logic. 0 = DISABLE 1 = ENABLE
23	DISABLE	CLK_ENB_UARTC: Enable clock to UARTC controller 0 = DISABLE 1 = ENABLE
22	DISABLE	CLK_ENB_I2C2: Enable clock to I2C2 controller. 0 = DISABLE 1 = ENABLE
20	DISABLE	CLK_ENB_CSI: Enable clock to CSI controller 0 = DISABLE 1 = ENABLE
19	DISABLE	CLK_ENB_HDMI: Enable clock to HDMI 0 = DISABLE 1 = ENABLE
18	DISABLE	CLK_ENB_HSI: Enable clock to MIPI baseband HSI controller. 0 = DISABLE 1 = ENABLE
16	DISABLE	CLK_ENB_DSI: Enable clock to DSI controller 0 = DISABLE 1 = ENABLE
15	DISABLE	CLK_ENB_ I2C5: Enable clock to I2C5 controller 0 = DISABLE 1 = ENABLE
14	DISABLE	CLK_ENB_SPI3: Enable clock to SPI3 controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
12	DISABLE	CLK_ENB_SPI2: Enable clock to SPI2 Controller. 0 = DISABLE 1 = ENABLE
11	ENABLE	CLK_ENB_JTAG2TBC: Enable clock to jtag2tbc Interface. 0 = DISABLE 1 = ENABLE
10	ENABLE	CLK_ENB_SNOR: Enable clock to NOR Flash Controller. 0 = DISABLE 1 = ENABLE
9	DISABLE	CLK_ENB_SPI1: Enable clock to SPI1 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

5.7.9 CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x01f02a00 (0b00000x011111000x00101x1000000x0x)

Bit	Reset	Description
31	DISABLE	CLK_ENB_XUSB_DEV: Enable clock to XUSB DEV. 0 = DISABLE 1 = ENABLE
30	DISABLE	CLK_ENB_DEV1_OUT: Enable clock to DEV1 pad. 0 = DISABLE 1 = ENABLE
29	DISABLE	CLK_ENB_DEV2_OUT: Enable clock to DEV2 pad. 0 = DISABLE 1 = ENABLE
28	DISABLE	CLK_ENB_SUS_OUT: Enable clock to SUS pad. 0 = DISABLE 1 = ENABLE
27	DISABLE	CLK_ENB_MSENC: Enable MSENC clk. 0 = DISABLE 1 = ENABLE
26	ENABLE	CLK_M_DOUBLER_ENB: Enable CLK_M clock doubler
25	DISABLE	CLK_ENB_XUSB_HOST: Enable clock to XUSB HOST 0 = DISABLE 1 = ENABLE
24	ENABLE	CLK_ENB_CRAM2: Enable COP cache RAM clock. 0 = DISABLE 1 = ENABLE
23	ENABLE	CLK_ENB_IRAMD: Enable IRAMD clock. 0 = DISABLE 1 = ENABLE
22	ENABLE	CLK_ENB_IRAMC: Enable IRAMC clock. 0 = DISABLE 1 = ENABLE
21	ENABLE	CLK_ENB_IRAMB: Enable IRAMB clock. 0 = DISABLE 1 = ENABLE
20	ENABLE	CLK_ENB_IRAMA: Enable IRAMA clock. 0 = DISABLE 1 = ENABLE
19	DISABLE	CLK_ENB_TSEC: Enable TSEC clock. 0 = DISABLE 1 = ENABLE
18	DISABLE	CLK_ENB_DSIB: Enable clock to DSIB 0 = DISABLE 1 = ENABLE
17	DISABLE	CLK_ENB_I2C_SLOW: Enable clock to I2C_SLOW 0 = DISABLE 1 = ENABLE
15	DISABLE	CLK_ENB_DTV: Enable clock to DTV controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
14	DISABLE	CLK_ENB_SOC_THERM: Enable clock to SOC_THERM controller. 0 = DISABLE 1 = ENABLE
13	ENABLE	CLK_ENB_TRACECLKIN: Enable traceclkin to CoreSight. 0 = DISABLE 1 = ENABLE
11	ENABLE	CLK_ENB_AVPUCQ: Enable clock to AVPUCQ. 0 = DISABLE 1 = ENABLE
9	ENABLE	CLK_ENB_CSITE: Enable clock to CoreSight. 0 = DISABLE 1 = ENABLE
8	DISABLE	CLK_ENB_AFI: Enable clock to AFI. 0 = DISABLE 1 = ENABLE
7	DISABLE	CLK_ENB_OWR: Enable clock to OWR. 0 = DISABLE 1 = ENABLE
6	DISABLE	CLK_ENB_PCIE: Enable clock to PCIe. 0 = DISABLE 1 = ENABLE
5	DISABLE	CLK_ENB_SDMMC3: Enable clock to SDMMC3. 0 = DISABLE 1 = ENABLE
4	DISABLE	CLK_ENB_SPI4: Enable clock to SPI4. 0 = DISABLE 1 = ENABLE
3	DISABLE	CLK_ENB_I2C3: Enable clock to I2C3. 0 = DISABLE 1 = ENABLE
1	DISABLE	CLK_ENB_UARTD: Enable clock to UARTD. 0 = DISABLE 1 = ENABLE

5.7.10 CLK_RST_CONTROLLER_CCLK_BURST_POLICY_0

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

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

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

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

- Software can simply keep changing the clock source to one state (i.e., just CWAKEUP_IDLE_SOURCE) without changing the CPU_STATE field.

- Software can have the concept of multiple states by first setting up CWAKEUP__SOURCE and then changing the CPU_STATE field.
- Software can enable hardware auto detect of IRQ/FIQ to jump to the IRQ or FIQ state.

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

CCLK and SCLK Super Clock Divider Control

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

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

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

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

Offset: 0x20 │ Read/Write: R/W │ Reset: 0x10X00000 (0b00010000xxxxxxx00000000000000000)

Bit	R/W	Reset	Description
31:28	RW	0x1	CPU_STATE: 0000=32 kHz Clock source; 0001=IDLE Clock Source; 001X=Run clock source; 01XX=IRQ Clock Source; 1XXX=FIQ Clock Source 0 = STDBY 1 = IDLE 2 = RUN 4 = IRQ 8 = FIQ
27	RW	0x0	COP_AUTO_CWAKEUP_FROM_FIQ: 0 = NOP ; 1=Burst on COP FIQ
26	RW	0x0	CPU_AUTO_CWAKEUP_FROM_FIQ: 0 = NOP ; 1=Burst on CPU FIQ
25	RW	0x0	COP_AUTO_CWAKEUP_FROM_IRQ: 0 = NOP ; 1=Burst on COP IRQ
24	RW	0x0	CPU_AUTO_CWAKEUP_FROM_IRQ: 0 = NOP ; 1=Burst on CPU IRQ
23	RO	X	TSENSOR_SLOWDOWN: 0 = Normal; 1 = cpug_clk/2 triggered by temperature sensor.
16	RW	0x0	CCLK_RESERVED: Reserved. CPULP uses this bit for div2 bypass.

Bit	R/W	Reset	Description
15:12	RW	0x0	CWAKEUP_FIQ_SOURCE: 0000 = clk_m, 0001 = pllC_out0, 0010 = clk_s, 0011 = pllM_out0, 0100 = pllP_out0, 0101 = pllP_out4, 0110 = pllC2_out0, 0111 = pllC3_out0, 1000 = PLLX_out0 (low jitter), 1001 = reserved, 1110 = PLLX_out0, (low jitter), 1111 = reserved 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = RESERVED9 14 = PLLX_OUT0 15 = RESERVED15
11:8	RW	0x0	CWAKEUP_IRQ_SOURCE: Same definitions as CWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = RESERVED9 14 = PLLX_OUT0_LJ 15 = RESERVED15
7:4	RW	0x0	CWAKEUP_RUN_SOURCE: Same definitions as CWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = RESERVED9 14 = PLLX_OUT0 15 = RESERVED15
3:0	RW	0x0	CWAKEUP_IDLE_SOURCE: Same definitions as CWAKEUP_FIQ 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = RESERVED9 14 = PLLX_OUT0 15 = RESERVED15

5.7.11 CLK_RST_CONTROLLER_SUPER_CCLK_DIVIDER_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

Bit	Reset	Description
31	DISABLE	SUPER_CDIV_ENB: 0 = Disable divider. 0 = DISABLE 1 = ENABLE
30	DISABLE	SUPER_CDIV_USE_THERM_CONTROLS: 1 = Use therm controls for pulse skipper (cpug only) 0 = DISABLE 1 = ENABLE
28	DISABLE	CCLK_INVERT_DCD: 1 = Enable inversion of DCD (duty-cycle distortion), 0 = Disable inversion of DCD 0 = DISABLE 1 = ENABLE See “Invert Duty-Cycle Distortion Control” in this section for more information.
27	0x0	SUPER_CDIV_DIS_FROM_COP_FIQ: 0 = enable FIQ, disable FIQ.
26	0x0	SUPER_CDIV_DIS_FROM_CPU_FIQ: 0 = enable FIQ, disable FIQ.
25	0x0	SUPER_CDIV_DIS_FROM_COP_IRQ: 0 = enable IRQ, disable IRQ.
24	0x0	SUPER_CDIV_DIS_FROM_CPU_IRQ: 0 = enable IRQ, disable IRQ.
23:16	0x0	CCLK_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_CDIV_DIVISOR: Actual value = n + 1.

5.7.12 CLK_RST_CONTROLLER_SCLK_BURST_POLICY_0

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

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

Bit	Reset	Description
14:12	0x0	SWAKEUP_FIQ_SOURCE: 000 = clk_m, 001 = pllC_out1, 010 = pllP_out4, 011 = pllP_out0, 100 = pllP_out2, 101 = pllC_out0, 110 = clk_s, 111 = pllM_out1, 0 = CLKM 1 = PLLC_OUT1 2 = PLLP_OUT4 3 = PLLP_OUT0 4 = PLLP_OUT2 5 = PLLC_OUT0 6 = CLKS 7 = PLLM_OUT1
10:8	0x0	SWAKEUP_IRQ_SOURCE: Same definitions as SWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT1 2 = PLLP_OUT4 3 = PLLP_OUT0 4 = PLLP_OUT2 5 = PLLC_OUT0 6 = CLKS 7 = PLLM_OUT1
6:4	0x0	SWAKEUP_RUN_SOURCE: Same definitions as SWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT1 2 = PLLP_OUT4 3 = PLLP_OUT0 4 = PLLP_OUT2 5 = PLLC_OUT0 6 = CLKS 7 = PLLM_OUT1
2:0	0x0	SWAKEUP_IDLE_SOURCE: Same definitions as SWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT1 2 = PLLP_OUT4 3 = PLLP_OUT0 4 = PLLP_OUT2 5 = PLLC_OUT0 6 = CLKS 7 = PLLM_OUT1

5.7.13 CLK_RST_CONTROLLER_SUPER_SCLK_DIVIDER_0

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

Bit	Reset	Description
31	DISABLE	SUPER_SDIV_ENB: 0 = Disable divider. 0 = DISABLE 1 = ENABLE
27	0x0	SUPER_SDIV_DIS_FROM_COP_FIQ: 0 = enable FIQ, disable FIQ.
26	0x0	SUPER_SDIV_DIS_FROM_CPU_FIQ: 0 = enable FIQ, disable FIQ.
25	0x0	SUPER_SDIV_DIS_FROM_COP_IRQ: 0 = enable IRQ, disable IRQ.
24	0x0	SUPER_SDIV_DIS_FROM_CPU_IRQ: 0 = enable IRQ, disable IRQ.
15:8	0x0	SUPER_SDIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_SDIV_DIVISOR: Actual value = n + 1.

5.7.14 CLK_RST_CONTROLLER_CLK_SYSTEM_RATE_0

HCLK/PCLK

SCLK is the main system clock which can run up to 275 MHz.

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

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

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

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.

5.7.15 CLK_RST_CONTROLLER_CLK_MASK_ARM_0

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

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

5.7.16 CLK_RST_CONTROLLER_MISC_CLK_ENB_0

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

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

5.7.17 CLK_RST_CONTROLLER_CLK_CPU_CMPLX_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

The 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 for 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: However, since the CPU clock can be selected from 1-of-9 clock sources (refer to the CCLK_BURST_POLICY register), the user is responsible for not selecting a bridge divide-down CPU clock ratio that will exceed 1/4 of the "MAX" CPU frequency supported. The "MAX" CPU frequency is 1.1 GHz.

Offset: 0x4c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx0000xxxxxx00)

Bit	Reset	Description
11	0x0	CPU3_CLK_STP: 1 = CPU3 clock stop, 0 = CPU3 clock run.
10	0x0	CPU2_CLK_STP: 1 = CPU2 clock stop, 0 = CPU2 clock run.
9	0x0	CPU1_CLK_STP: 1 = CPU1 clock stop, 0 = CPU1 clock run.
8	0x0	CPU0_CLK_STP: 1 = CPU0 clock stop, 0 = CPU0 clock run.
1:0	0x0	CPU_BRIDGE_CLKDIV: Unused but available.

5.7.18 CLK_RST_CONTROLLER_OSC_CTRL_0

Oscillator Control

"osc" can have any of the hardware-supported frequencies (12, 13, 19.2, 26, 16.8, 38.4, 48 MHz) . The OSC_FREQ field provides a way for software to inform hardware what the incoming clock frequency is. This information is used by hardware to auto setup the parameters (DIVN, DIVM, DIVP, CPCON, LFCON, VCOCON, DCCON, OUT1_RATIO, OUT2_RATIO, OUT3_RATIO, OUT4_RATIO, and OUT5_RATIO) to PLLP and its dividers. If a different frequency is used, there is a way to override the hardware auto-generated PLLP parameters.

Offset: 0x50 │ Read/Write: R/W │ Reset: 0x800003f1 (0b10000000000000x00000xx111111x0x1) | SW Default: 0x00000010

Bit	Reset	SW Default	Description
31:28	0x0	NONE	OSC_FREQ: 0000 = 13.0 MHz, 0100 = 19.2 MHz, 1000 = 12.0 MHz, 1100 = 26.00 MHz, 0001 = 16.8 MHz, 0101 = 38.4 MHz*, 1001 = 48.0 MHz*. *Set PLL_REF_DIV to /2 for 38.4 MHz and /4 for 48 MHz. Unused code map to 13 MHz setting in hardware. 0 = OSC13 4 = OSC19P2 8 = OSC12 12 = OSC26 1 = OSC16P8 5 = OSC38P4 9 = OSC48
27:26	0x0	NONE	PLL_REF_DIV: PLL reference clock divide. 00 = /1 01 = /2 10 = /4 0 = DIV1 1 = DIV2 2 = DIV4 3 = RESERVED

Bit	Reset	SW Default	Description
25:18	0x0	NONE	OSCFI_SPARE: Crystal oscillator spare register control.
16:12	0x0	NONE	XODS: Crystal oscillator duty cycle control.
9:4	0x3f	0x1	XOFS: Crystal oscillator drive strength control.
2	0x0	NONE	CLK_OK: Crystal oscillator clk_ok.
0	0x1	NONE	XOE: Crystal oscillator enable (1 = enable).

5.7.19 CLK_RST_CONTROLLER_PLL_LFSR_0

Offset: 0x54 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

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

5.7.20 CLK_RST_CONTROLLER_OSC_FREQ_DET_0

Oscillator Frequency Detect

The “osc” can have any one of the seven supported crystal frequencies (12, 13, 19.2, 26, 16.8, 38.4, 48 MHz). Hardware provides the following mechanism to detect which frequency of osc is running. To determine the osc frequency, program the number of 32.768 kHz clock periods to create a fixed time window in which the "osc" clocks will be counted. Once the counting process is done, the count value can be used to determine the osc frequency.

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

Offset: 0x58 │ Read/Write: R/W │ Reset: 0x00000000 (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: Indicates the number of 32.768 kHz clock periods as window in n+1 scheme.

5.7.21 CLK_RST_CONTROLLER_OSC_FREQ_DET_STATUS_0

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

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

5.7.22 CLK_RST_CONTROLLER_PLLE_SS_CNTL_0

Offset: 0x68 | Read/Write: R/W | Reset: 0x00005c00 (0b00000000000000000010111x000000000) | Default: 0x20010025

Bit	Reset	SW Default	Description
31:30	0x0	_NONE_	PLLE_INTEGOFFSET: interpolator bias current.
29:24	0x0	0x20	PLLE_SSCINCINTRV: Triangle generator increment interval control.
23:16	0x0	0x1	PLLE_SSCINC: Triangle generator increment control.
15	0x0	0x0	PLLE_SSCINVERT: 0: gives down spread, 1: gives up-spread.
14	0x1	0x0	PLLE_SSCCENTER: 0: gives control to SSCINVERT, 1: enables center spread.
13	0x0	_NONE_	PLLE_SSCPDMBYP: Bypass from pulse density modulator. Normally set to zero.
12	0x1	0x0	PLLE_SSCBYP: 0: enables spreading, 1: disables spreading.
11	0x1	0x0	PLLE_INTERP_RESET: Interpolator reset. 0: normal operation, 1: resets SS machine.
10	0x1	0x0	PLLE_BYPASS_SS: When set feedback clock bypasses interpolator. Default value is zero.
8:0	0x0	0x25	PLLE_SSCMAX: Spread limit control.

5.7.23 CLK_RST_CONTROLLER_PLLC_BASE_0

Tegra K1 devices have 14 PLLs controllable by clock control. Consult the PLL specifications for electrical requirements. Below are the PLL cells used for each PLL referenced in this TRM:

- PLLC is of type PLL14G_DYN_ESD.
- PLLC2 and PLLC3 are of type PLL12G_DIG_E1.
- PLLM is of type PLL10G_4PH_ESD.
- PLLP and PLLA are of type CLKPLL700_LP.
- PLLU is of type CLKPLL960_USB.
- PLLD is of type CLKPLL15G_MIPI.
- PLLD2 is of type PLL12G_SSD_ESD.
- PLLDP and PLLC4 are of type PLL12G_SSD_AESD
- PLLX is of type PLL24G_DYN_PRB_ESD.
- refPLLE is of type PLL600_ESD.
- PLLE is of type PLL24G_SSA_33VCML_ESD.

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

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

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

Crystals 38.4 MHz and 48 MHz require pre-divider PLL_REF_DIV settings of 2 and 4, respectively, to meet the input frequency requirements of all PLLs that use "osc_div_clk" as a reference. All other supported crystals are used by the PLLs directly, without any pre-divider.

Contact your NVIDIA representative for more details on the PLLs.

PLLP Configuration Information

PLLP is configured as a fixed frequency PLL. The code in the Boot ROM configures PLLP at a fixed 408 MHz. Later on, software (typically the Boot Loader) may change this to another "fixed" frequency.

For Tegra K1 devices, software changes this frequency from 216 MHz to 408 MHz. Boot ROM configuration values for 408 MHz are listed below.

Table 20: Boot ROM Configuration to 216 MHz

Reference Frequency	13.0 MHz	19.2 MHz	12.0 MHz	26.0 MHz	16.8MHz	38.4MHz	48.0MHz	Resulting Frequency
DIVN	408 (198h)	340 (154h)	408 (198h)	204 (0cch)	340 (154h)	340 (154h)	204 (0cch)	-
DIVM	13 (0dh)	16 (10h)	12 (0ch)	13 (0dh)	14 (0eh)	16 (10h)	6 (06h)	-
DIVP	1 (0h)	1 (0h)	1 (0h)	1 (0h)	1 (0h)	1 (0h)	1 (0h)	-
CPCON	8 (8h)	4 (4h)	8 (8h)	8 (8h)	4 (4h)	4 (4h)	8 (8h)	-
LFCON	0 (0h)	0 (0h)	0 (0h)	0 (0h)	0 (0h)	0 (0h)	0 (0h)	-
VCOCON	0 (0h)	0 (0h)	0 (0h)	0 (0h)	0 (0h)	0 (0h)	0 (0h)	-
DCCON	0 (0b)	0 (0b)	0 (0b)	0 (0b)	0 (0b)	0 (0b)	0 (0b)	-
PLL_REF_DIV					/1 (0h)	/2 (1h)	/4 (2h)	-
OUT1 div ratio	42.5 (53h)	42.5 (53h)	42.5 (53h)	42.5 (53h)	42.5 (53h)	42.5 (53h)	42.5 (53h)	9.6 MHz
OUT2 div ratio	2.0 (02h)	2.0 (02h)	2.0 (02h)	2.0 (02h)	2.0 (02h)	2.0 (02h)	2.0 (02h)	204.0 MHz
OUT3 div ratio	4.0 (06h)	4.0 (06h)	4.0 (06h)	4.0 (06h)	4.0 (06h)	4.0 (06h)	4.0 (06h)	102.0 MHz
OUT4 div ratio	4.0 (06h)	4.0 (06h)	4.0 (06h)	4.0 (06h)	4.0 (06h)	4.0 (06h)	4.0 (06h)	102.0 MHz
OUT5 div ratio	2.0 (02h)	2.0 (02h)	2.0 (02h)	2.0 (02h)	2.0 (02h)	2.0 (02h)	2.0 (02h)	204.0 MHz

Table 21: PLLU Configuration Information (Reference Clock osc_div_clk and PLLU-FOs fixed at 12 MHz/48 MHz/60 MHz/480 MHz)

Reference Frequency	13.0 MHz	19.2 MHz	12.0 MHz	26.0 MHz	16.8 MHz	38.4MHz	48.0MHz
DIVN	960 (3c0h)	200 (0c8h)	960 (3c0h)	960 (3c0h)	400 (190h)	200 (0c8h)	960 (3c0h)
DIVM	13 (0dh)	4 (04h)	12 (0ch)	26 (1ah)	7 (07h)	4 (04h)	12 (0ch)
CPCON	12 (ch)	3 (3h)	12 (ch)	12 (ch)	5 (5h)	3 (3h)	12 (ch)
LFCON	2 (2h)	2 (2h)	2 (2h)	2 (2h)	2 (2h)	2 (2h)	2 (2h)
PLL_REF_DIV					/1 (0h)	/2 (1h)	/4 (2h)

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

PLLX is shared between G and LP CPU clusters. To save power, the proper PLLX_FO_LP_DISABLE or PLLX_FO_G_DISABLE bit should only be de-asserted when the output is being used.

Offset: 0x80 │ Read/Write: R/W │ Reset: 0x0000010c (0bx000xxxx0000xxxx0000000100001100)

Bit	R/W	Reset	Description
30	RW	DISABLE	PLLC_ENABLE: 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
29	RW	REF_ENABLE	PLLC_REF_DIS: 0 = enable reference clock, 1 = disable reference clock. 0 = REF_ENABLE 1 = REF_DISABLE
28	RW	0x0	PLLC_LOCK_OVERRIDE
27	RO	X	PLLC_LOCK: 0 = not lock, 1 = lock.
23:20	RW	0x0	PLLC_DIVP: 0 = post divider (2^n).
15:8	RW	0x1	PLLC_DIVN: PLL feedback divider.
7:0	RW	0xc	PLLC_DIVM: PLL input divider.

5.7.24 CLK_RST_CONTROLLER_PLLC_OUT_0

Offset: 0x84 │ Read/Write: R/W │ Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxx00000000xxxxxx10)

Bit	Reset	Description
15:8	0x0	PLLC_OUT1_RATIO: PLLC_OUT1 divider from base PLLC (lsb denoted 0.5x).
1	ENABLE	PLLC_OUT1_CLKEN: PLLC_OUT1 divider clk enable. 0 = DISABLE 1 = ENABLE
0	RESET_ENABLE	PLLC_OUT1_RSTN: PLLC_OUT1 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

5.7.25 CLK_RST_CONTROLLER_PLLC_MISC2_0

Offset: 0x88 │ Read/Write: R/W │ Reset: 0x0000000X (0bxx00000000000000000000000000000x)

Bit	R/W	Reset	Description
29:28	RW	0x0	PLLC_PTS: PTS field for PLLC. 0 = DISABLE 1 = FO 2 = VCO
27	RW	0x0	PLLC_EN_FSTLCK
26	RW	0x0	PLLC_CLAMP_NDIV
25	RW	0x0	PLLC_EN_DYNRAMP
24:17	RW	0x0	PLLC_DYNRAMP_STEPA
16:9	RW	0x0	PLLC_DYNRAMP_STEPB
8:1	RW	0x0	PLLC_NDIV_NEW
0	RO	X	PLLC_DYNRAMP_DONE

5.7.26 CLK_RST_CONTROLLER_PLLC_MISC_0

Offset: 0x8c │ Read/Write: R/W │ Reset: 0x0X000000 (0bx00001x00000000000000000000000)

Bit	R/W	Reset	Description
30	RW	0x0	PLLC_OUT1_DIV_BYP: 1 = bypass PLLC_OUT1 divider.
29:28	RW	0x0	PLLC_KCP

Bit	R/W	Reset	Description
27	RW	0x0	PLLC_KVCO
26	RW	0x1	PLLC_IDDQ
25	RO	X	PLLC_FREQ_LOCK
24	RW	0x0	PLLC_LOCK_ENABLE: 0 = DISABLE 1 = ENABLE
23:0	RW	0x0	PLLC_SETUP: Base PLLC VCO range setup control.

5.7.27 CLK_RST_CONTROLLER_PLLM_BASE_0

Offset: 0x90 │ Read/Write: R/W │ Reset: 0x00000801 (0b000xxxxxxxxxxxx00000100000000001)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLM_BYPASSPLL: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLM_ENABLE: 0 = disable, 1 = enable. (Will invert and connect to IDDQ) 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLM_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLM_LOCK: 0 = not lock, 1 = lock. Phase and frequency
26	RO	X	PLLM_FREQ_LOCK: 0 = not lock, 1 = lock. Frequency lock only
20	RW	0x0	PLLM_DIV2: 1= Frequency of CLKOUT*=VCOCLK/2
19	RW	0x0	PLLM_CLAMP_PX: 1 = clamp CLKOUT_P* / CLKOUT_P*B to 0/1 has priority over IDDQ and ENABLE and PLLAVDD **Note**: PLLM_CLAMP_PX will be cleared before using PLLM. Its purpose is to avoid excessive CDB power consumption while analog PLL power is coming up and the differential output of PLLM is still stabilizing.
15:8	RW	0x8	PLLM_DIVN: PLL feedback divider.
7:0	RW	0x1	PLLM_DIVM: PLL input divider.

5.7.28 CLK_RST_CONTROLLER_PLLM_OUT_0

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

Offset: 0x94 │ Read/Write: R/W │ Reset: 0x00000002 (0bxxxxxxxxxxxxxxxx000000000xxxxxx10)

Bit	Reset	Description
16	0x0	PLLM_OUT1_DIV_BYP: 1 = bypass PLLM_OUT1 divider
15:8	0x0	PLLM_OUT1_RATIO: PLLM_OUT1 divider from base PLLM (lsb denotes 0.5x).
1	ENABLE	PLLM_OUT1_CLKEN: PLLM_OUT1 divider clk enable. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	RESET_ENABLE	PLLM_OUT1_RSTN: PLLM_OUT1 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

5.7.29 CLK_RST_CONTROLLER_PLLM_MISC1_0

Offset: 0x98 | Read/Write: R/W | Reset: 0x00000000 (0bx00000000000000000000000000000000)

Bit	Reset	Description
30	0x0	PLLM_PD_LSHIFT_PH135
29	0x0	PLLM_PD_LSHIFT_PH90
28	0x0	PLLM_PD_LSHIFT_PH45
27	0x0	PLLM_CLAMP_PH135
26	0x0	PLLM_CLAMP_PH90
25	0x0	PLLM_CLAMP_PH45
24	0x0	PLLM_CLAMP_PH0
23:0	0x0	PLLM_SETUP: SETUP fields

5.7.30 CLK_RST_CONTROLLER_PLLM_MISC2_0

Offset: 0x9c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx000x0000000)

Bit	Reset	Description
10	0x0	PLLM_ENABLE_SW_OVERRIDE: Let software override values on clamp/bypass versus hardware FSM control
9:8	0x0	PLLM_PTS: PTS field for external mux. 0 = DISABLE 1 = FO
6	0x0	PLLM_EN_FSTLCK
5	0x0	PLLM_IDDQ
4	DISABLE	PLLM_PD_LCKDET:1 = Power down lock detect 0 = DISABLE 1 = ENABLE
3	0x0	PLLM_LOCK_OVERRIDE: Forces lock to 1
2:1	0x0	PLLM_KCP: Charge Pump Gain control
0	0x0	PLLM_KVCO:VCO gain

5.7.31 CLK_RST_CONTROLLER_PLLP_BASE_0

Offset: 0xa0 | Read/Write: R/W | Reset: 0x0X00010c (0b0000xxxxx000xx0000000001xxx01100)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLP_BYPASS: 0 = no bypass 1 = bypass. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
30	RW	DISABLE	PLLP_ENABLE: 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLP_REF_DIS: 0 = enable reference clock 1 = disable reference clock 0 = REF_ENABLE 1 = REF_DISABLE
28	RW	DISABLE	PLLP_BASE_OVRRIDE: 0 = disallow base override 1 = allow base override 0 = DISABLE 1 = ENABLE
27	RO	X	PLLP_LOCK: 0 = not lock, 1 = lock.
22:20	RW	0x0	PLLP_DIVP: 0 = post divider (2^n).
17:8	RW	0x1	PLLP_DIVN: PLL feedback divider.
4:0	RW	0xc	PLLP_DIVM: PLL input divider.

5.7.32 CLK_RST_CONTROLLER_PLLP_OUTA_0

Offset: 0xa4 │ Read/Write: R/W │ Reset: 0x00020002 (0b00000000xxxxx01000000000xxxxx010)

Bit	Reset	Description
31:24	0x0	PLLP_OUT2_RATIO: PLLP_OUT2 divider from base PLLP (lsb denotes 0.5x).
18	DISABLE	PLLP_OUT2_OVRRIDE: 0 = disallow PLLP_OUT2 ratio override 1 = enable override 0 = DISABLE 1 = ENABLE
17	ENABLE	PLLP_OUT2_CLKEN: PLLP_OUT2 divider clock enable. 0 = DISABLE 1 = ENABLE
16	RESET_ENABLE	PLLP_OUT2_RSTN: PLLP_OUT2 divider reset. 0 = reset 1 = not reset 0 = RESET_ENABLE 1 = RESET_DISABLE
15:8	0x0	PLLP_OUT1_RATIO: PLLP_OUT1 divider from base PLLP (lsb denotes 0.5x).
2	DISABLE	PLLP_OUT1_OVRRIDE: 0 = Disallow PLLP_OUT1 ratio override 1 = enable override 0 = DISABLE 1 = ENABLE
1	ENABLE	PLLP_OUT1_CLKEN: PLLP_OUT1 divider clock enable. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	RESET_ENABLE	PLLP_OUT1_RSTN: PLLP_OUT1 divider reset. 0 = reset 1 = not reset 0 = RESET_ENABLE 1 = RESET_DISABLE

5.7.33 CLK_RST_CONTROLLER_PLLP_OUTB_0

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

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

Bit	Reset	Description
31:24	0x0	PLLP_OUT4_RATIO: PLLP_OUT4 divider from base PLLP (lsb denotes 0.5x).
18	DISABLE	PLLP_OUT4_OVRRIDE: 0 = disallow PLLP_OUT4 ratio override 1 = enable override 0 = DISABLE 1 = ENABLE
17	ENABLE	PLLP_OUT4_CLKEN: PLLP_OUT4 divider clock enable. 0 = DISABLE 1 = ENABLE
16	RESET_ENABLE	PLLP_OUT4_RSTN: PLLP_OUT4 divider reset. 0 = reset 1 = not reset 0 = RESET_ENABLE 1 = RESET_DISABLE
15:8	0x0	PLLP_OUT3_RATIO: PLLP_OUT3 divider from base PLLP (lsb denotes 0.5x).
2	DISABLE	PLLP_OUT3_OVRRIDE: 0 = Disallow PLLP_OUT3 ratio override. 0 = DISABLE 1 = ENABLE
1	ENABLE	PLLP_OUT3_CLKEN: PLLP_OUT3 divider clock enable. 0 = DISABLE 1 = ENABLE
0	RESET_ENABLE	PLLP_OUT3_RSTN: PLLP_OUT3 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

5.7.34 CLK_RST_CONTROLLER_PLLP_MISC_0

Offset: 0xac │ Read/Write: R/W │ Reset: 0x00040100 (0bxxxx00000000x10000000000100000000)

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.

Bit	Reset	Description
21	0x0	PLLP_CLKEN_FVCO: FVCO output clock enable
20	0x0	PLLP_DCCON: Base PLLP DCCON control.
18	0x0	PLLP_LOCK_ENABLE: 0 = DISABLE 1 = ENABLE
17:12	0x0	PLLP_LOCK_SEL: Lock select.
11:8	0x1	PLLP_CPCON: Base PLLP charge pump setup control.
7:4	0x0	PLLP_LFCON: Base PLLP loop filter setup control.
3:0	0x0	PLLP_VCOCON: Base PLLP VCO range setup control.

5.7.35 CLK_RST_CONTROLLER_PLLA_BASE_0

Offset: 0xb0 │ Read/Write: R/W │ Reset: 0x0X00010c (0b000xxxxxx000xx0000000001xxx01100)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLA_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLA_ENABLE: 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLA_REF_DIS: 0 = enable reference clock 1 = disable reference clock 0 = REF_ENABLE 1 = REF_DISABLE
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.

5.7.36 CLK_RST_CONTROLLER_PLLA_OUT_0

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

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

Bit	Reset	Description
15:8	0x0	PLLA_OUT0_RATIO: PLLA_OUT0 divider from base PLLA (lsb denotes 0.5x). 0 = DISABLE 1 = ENABLE
1	ENABLE	PLLA_OUT0_CLKEN: PLLA_OUT0 divider clk enable. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	RESET_ENABLE	PLLA_OUT0_RSTN: PLLA_OUT0 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

5.7.37 CLK_RST_CONTROLLER_PLLA_MISC_0

Offset: 0xbc │ Read/Write: R/W │ Reset: 0x00000100 (0bx0xxxxxx0000x00000000100000000)

Bit	Reset	Description
30	0x0	PLLA_OUT0_DIV_BYP: 1 = bypass PLLA_OUT0 divider.
23:22	0x0	PLLA_PTS: Base PLLA test output select.
21	0x0	PLLA_CLKEN_FVCO: FVCO output clock enable
20	0x0	PLLA_DCCON: Base PLLA DCCON control.
18	0x0	PLLA_LOCK_ENABLE: 0 = DISABLE 1 = ENABLE
17:12	0x0	PLLA_LOCK_SEL: Lock select.
11:8	0x1	PLLA_CPCON: Base PLLA charge pump setup control.
7:4	0x0	PLLA_LFCON: Base PLLA loop filter setup control.
3:0	0x0	PLLA_VCOCON: Base PLLA VCO range setup control.

5.7.38 CLK_RST_CONTROLLER_PLLU_BASE_0

Offset: 0xc0 │ Read/Write: R/W │ Reset: 0x0X00010c (0b000xxx010000xx0000000001xxx01100)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLU_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLU_ENABLE: This bit is used only when the PLLU_OVERRIDE bit is set. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLU_REF_DIS: 0 = enable reference clock 1 = disable reference clock 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLU_LOCK: 0 = not lock, 1 = lock.
25	RW	0x0	PLLU_CLKENABLE_48M: FO_48M output enable. This bit is use only when the PLLU_OVERRIDE bit is set.
23	RW	0x0	PLLU_CLKENABLE_ICUSB: FO_ICUSB output enable. This bit is used only when the PLLU_OVERRIDE bit is set.

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

5.7.39 CLK_RST_CONTROLLER_PLLU_MISC_0

Offset: 0xcc │ Read/Write: R/W │ Reset: 0x00407100 (0bxx000xxxx1xxxx000111000100000000)

Bit	Reset	Description
29:27	0x0	PLLU_PTS: Base PLLU test output select.
22	0x1	PLLU_LOCK_ENABLE: 0 = DISABLE 1 = ENABLE
17:12	0x7	PLLU_LOCK_SEL: Lock select.
11:8	0x1	PLLU_CPCON: Base PLLU charge pump setup control.
7:4	0x0	PLLU_LFCON: Base PLLU loop filter setup control.
3:0	0x0	PLLU_VCOCON: Base PLLU VCO range setup control.

5.7.40 CLK_RST_CONTROLLER_PLLD_BASE_0

Offset: 0xd0 │ Read/Write: R/W │ Reset: 0x0X00010c (0b000xx00x0000x00000000001xxx01100)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLD_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLD_ENABLE: 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLD_REF_DIS: 0 = Enable reference clock, 1 = Disable reference clock. 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLD_LOCK: 0 = not lock, 1 = lock.
26	RW	0x0	PLLD_CLKENABLE_CSI: 0 = disable, 1 = normal operation. Enable CSI fast and slow P/N clocks to pad macros

Bit	R/W	Reset	Description
25	RW	PLL_D	DSIA_CLK_SRC: Only PLLD is the source because it is only MIPI PLL. 0 = PLL_D 1 = PLL_D1
23	RW	0x0	CSI_CLK_SRC: 0 = Brick, 1 = PLL_D.
22:20	RW	0x0	PLLD_DIVP: 0 = Post divider (2^n).
18:8	RW	0x1	PLLD_DIVN: PLL feedback divider.
4:0	RW	0xc	PLLD_DIVM: PLL input divider.

5.7.41 CLK_RST_CONTROLLER_PLLD_MISC_0

Offset: 0xdc │ Read/Write: R/W │ Reset: 0x00000100 (0b00000000000000000000000100000000)

Bit	Reset	Description
31	0x0	PLLD_FO_MODE: 1 = 5-125 MHz, 0 = 40-1000 MHz.
30	0x0	PLLD_CLKENABLE: 0 = disable, 1 = normal operation.
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: 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.

5.7.42 CLK_RST_CONTROLLER_PLLX_BASE_0

Bypass is not supported. This field is kept for legacy purposes.

Offset: 0xe0 │ Read/Write: R/W │ Reset: 0x0X00010c (0b000xxxxx0000xxxx0000000100001100)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLX_BYPASS: PLLX_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLX_ENABLE: 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLX_REF_DIS: 0 = Enable reference clock, 1 = Disable reference clock. 0 = REF_ENABLE 1 = REF_DISABLE

Bit	R/W	Reset	Description
27	RO	X	PLLX_LOCK: 0 = Not lock, 1 = Lock.
23:20	RW	0x0	PLLX_DIVP: 0 = Post divider (2^n).
15:8	RW	0x1	PLLX_DIVN: PLL feedback divider.
7:0	RW	0xc	PLLX_DIVM: PLL input divider.

5.7.43 CLK_RST_CONTROLLER_PLLX_MISC_0

Note: PLLX source to cpulp (PLLX_OUT0) is divided by 2 by default. DIV2 is controlled by the PLLX_DIV2_BYPASS_LP bit. See the warning about DIV2 programming in the CLK_RST_CONTROLLER_CCLKLP_BURST_POLICY_0 register.

Offset: 0xe4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxx00xxxx00xxx0xxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29	0x0	PLLX_FO_LP_DISABLE: PLLX FO_LP output disable. 0=ON, 1=OFF
28	0x0	PLLX_FO_G_DISABLE: PLLX FO_G output disable. 0=ON, 1=OFF
23:22	0x0	PLLX_PTS: Base PLLX test output select. 0 = DISABLE 1 = FO 2 = VCO
18	0x0	PLLX_LOCK_ENABLE: 0 = DISABLE 1 = ENABLE

5.7.44 CLK_RST_CONTROLLER_PLLE_BASE_0

Offset: 0xe8 │ Read/Write: R/W │ Reset: 0x0d00c801 (0bx00011010000000011001000000000001)

Bit	Reset	Description
30	DISABLE	PLLE_ENABLE: PLL enable. 0 = DISABLE 1 = ENABLE
29	0x0	PLLE_LOCK_OVERRIDE: Forces PLL_LOCK and PLL_FREQLOCK to 1.
28	0x0	PLLE_FDIV4B: 0 = vcoclk/4, 1 = vcoclk/2 clock to the interpolator logic. Normally set to 0.
27:24	0xd	PLLE_PLDIV_CML: Divider control for CLOCKOUT_CML/CLOCKOUTB_CML.
23:16	0x0	PLLE_EXT_SETUP_23_16: Base PLLE setup [19:16].
15:8	0xc8	PLLE_NDIV: Feedback divider.
7:0	0x1	PLLE_MDIV: Input divider.

5.7.45 CLK_RST_CONTROLLER_PLLE_MISC_0

Offset: 0xec │ Read/Write: R/W │ Reset: 0x0000XX00 (0b0000000000000000x11xx00000000000)

Bit	R/W	Reset	Description
31:16	RW	0x0	PLLE_SETUP: Base PLLE setup [15:0].

Bit	R/W	Reset	Description
14	RW	0x1	PLLE_IDDQ_SWCTL: 0 = The PLLE is put in IDDQ mode by hardware (SATA +AFI+CAR) signals.1 = The PLLE is put in IDDQ by software. 0 = OFF 1 = ON (default)
13	RW	0x1	PLLE_IDDQ_OVERRIDE_VALUE: 0 = The PLLE is powered up. 1 = Software can put the PLLE in IDDQ by setting this bit and PLLE_IDDQ_SWCTL 0 = OFF 1 = ON (default)
12	RO	X	PLLE_FREQLOCK: 0 = frequency acquisition not achieved, 1 = frequency acquisition occurred. PLLE_LOCK_ENABLE must be enabled.
11	RO	X	PLLE_LOCK: 0 = not lock (phase+frequency), 1 = lock (phase+frequency). PLLE_LOCK_ENABLE must be enabled.
10	RW	REF_ENABLE	PLLE_REF_DIS: 0 = enable reference clock 1 = disable reference clock 0 = REF_ENABLE 1 = REF_DISABLE
9	RW	0x0	PLLE_LOCK_ENABLE: 0 = DISABLE 1 = ENABLE
8	RW	0x0	PLLE_PTS: Bypass PLL (similar to PTO control of other PLLs). 0 = PTO is 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_VREG_BG_CTRL: Base PLLE VREG BG control.
3:2	RW	0x0	PLLE_VREG_CTRL: Base PLLE VREG control.
1	RW	DISABLE	PLLE_EN_FSTLCK: 0 = DISABLE 1 = ENABLE
0	RW	0x0	PLLE_KVCO: Base PLLE VCO gain.

5.7.46 CLK_RST_CONTROLLER_CLK_SOURCE_I2S1_0

Clock Configuration for Each Peripheral

In general, each block or module has a dedicated CLK_SOURCE_ register that provides clock source and clock divider control to that device. The clock source select is typically 2 bits long to select one of the four clock sources. The divider is typically 8 bits long, consisting of a 7-bit integer and a 1-bit fraction. Furthermore, depending on the specific module, the CLOCK_SOURCE_ register may contain additional control bits.

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

Special PLL Clock Usage by Certain Peripherals

As mentioned previously, each peripheral has a clock source select in its corresponding CLK_SOURCE_ register. There are a number of modules that take in an additional fixed PLL clock source for portion of their logic.

Listed below are the modules/logic and the fixed PLL clock source they use.

Module/Logic	Fixed 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

More information about clock sources and divider widths can be found under “Hardware Features” in this section.

Audio Sync Clock

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

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

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

Bit	Reset	Description
31:29	CLK_M	I2S1_CLK_SRC: 000 = pllA_out0 010 = audio SYNC_CLK 1x or 2x 100 = pllP_out0 110 = clk_m 0 = PLLA_OUT0 2 = SYNC_CLK 4 = PLLP_OUT0 6 = CLK_M
28	0x1	I2S1_MASTER_CLKEN: Reserved.
7:0	0x0	I2S1_CLK_DIVISOR: N = Divide by (n+1) (lsb denotes 0.5x)

5.7.47 CLK_RST_CONTROLLER_CLK_SOURCE_I2S2_0

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

Bit	Reset	Description

Bit	Reset	Description
31:29	CLK_M	I2S2_CLK_SRC: 000 = pllA_out0 010 = audio SYNC_CLK 1x or 2x 100 = pllP_out0 110 = clk_m 0 = PLLA_OUT0 2 = SYNC_CLK 4 = PLLP_OUT0 6 = CLK_M
28	0x1	I2S2_MASTER_CLKEN: Reserved.
7:0	0x0	I2S2_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.48 CLK_RST_CONTROLLER_CLK_SOURCE_SPDIF_OUT_0

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

Bit	Reset	Description
31:29	CLK_M	SPDIFOUT_CLK_SRC: 000 = pllA_out0 010 = audio SYNC_CLK 1x or 2x 100 = pllP_out0 110 = clk_m 0 = PLLA_OUT0 2 = SYNC_CLK 4 = PLLP_OUT0 6 = CLK_M
7:0	0x0	SPDIF_OUT_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.49 CLK_RST_CONTROLLER_CLK_SOURCE_SPDIF_IN_0

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

Bit	Reset	Description
31:29	PLLP_OUT0	SPDIF_IN_CLK_SRC: 000 = pllP_out0 001 = pllC2_out0 010 = pllC_out0 011 = pllC3_out0 100 = pllM_out0 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0
7:0	0x0	SPDIF_IN_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.50 CLK_RST_CONTROLLER_CLK_SOURCE_PWM_0

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

Bit	Reset	Description
31:29	CLK_M	PWM_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = clk_s, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = CLK_S 6 = CLK_M
7:0	0x0	PWM_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.51 CLK_RST_CONTROLLER_CLK_SOURCE_SPI2_0

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

Bit	Reset	Description
31:29	CLK_M	SPI2_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	SPI2_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.52 CLK_RST_CONTROLLER_CLK_SOURCE_SPI3_0

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

Bit	Reset	Description
31:29	CLK_M	SPI3_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	SPI3_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.53 CLK_RST_CONTROLLER_CLK_SOURCE_I2C1_0

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

Bit	Reset	Description
31:29	CLK_M	I2C1_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
15:0	0x0	I2C1_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 1.0x)

5.7.54 CLK_RST_CONTROLLER_CLK_SOURCE_ I2C5_0

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

Bit	Reset	Description
31:29	CLK_M	I2C5_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
15:0	0x0	I2C5_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 1.0x)

5.7.55 CLK_RST_CONTROLLER_CLK_SOURCE_SPI1_0

Offset: 0x134 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SPI1_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	SPI1_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.56 CLK_RST_CONTROLLER_CLK_SOURCE_DISP1_0

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

Bit	Reset	Description
31:29	CLK_M	DISP1_CLK_SRC: 000 = pllP_out0, 010 = pllD_out0, 100 = pllC_out0, 110 = clk_m, 001 = pllM_out0, 011 = pllA_out0, 101 = pllD2_out0, 111 = 1'b0 Non sequential mapping used to preserve 2 MSBs to match legacy source select. 0 = PLLP_OUT0 2 = PLLD_OUT0 4 = PLLC_OUT0 6 = CLK_M 1 = PLLM_OUT0 3 = PLLA_OUT0 5 = PLLD2_OUT0

5.7.57 CLK_RST_CONTROLLER_CLK_SOURCE_DISP2_0

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

Bit	Reset	Description
31:29	CLK_M	DISP2_CLK_SRC: 000 = pllP_out0, 010 = pllD_out0, 011 = pllA_out0, 100 = pllC_out0, 101 = pllD2_out0, 110 = clk_m, 111 = 1'b0 Non sequential mapping used to preserve 2 MSBs to match legacy source select. 0 = PLLP_OUT0 1 = PLLM_OUT0 2 = PLLD_OUT0 3 = PLLA_OUT0 4 = PLLC_OUT0 5 = PLLD2_OUT0 6 = CLK_M

5.7.58 CLK_RST_CONTROLLER_CLK_SOURCE_ISP_0

Offset: 0x144 | Read/Write: R/W | Reset: 0x00000000 (0b000xxx00xxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLM_OUT0	ISP_CLK_SRC: 000 = pllM_out0, 001 = pllC_out0, 010 = pllP_out0, 011 = pllA_out0, 100 = pllC2_out0, 101 = pllC3_out0, 110 = clk_m, 111 = pllC4_out0 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA_OUT0 4 = PLLC2_OUT0 5 = PLLC3_OUT0 6 = CLK_M 7 = PLLC4_OUT0
7:0	0x0	ISP_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.59 CLK_RST_CONTROLLER_CLK_SOURCE_VI_0

Offset: 0x148 | Read/Write: R/W | Reset: 0x00000000 (0b000xxx00xxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLM_OUT0	VI_CLK_SRC: 000 = pllM_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = pllA_out0, 111 = pllC4_out0 0 = PLLM_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = PLLA_OUT0 7 = PLLC4_OUT0
25	0x0	PD2VI_CLK_SEL: 0 = pd2vi_clk, 1 = vi_sensor_clk.
24	INTERNAL	VI_CLK_SEL: 0 = select internal clock, 1 = select external clock (pd2vi_clk). 0 = INTERNAL 1 = EXTERNAL
7:0	0x0	VI_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.60 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC1_0

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

Bit	Reset	Description
31:29	CLK_M	SDMMC1_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 101 = pllE_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 5 = PLLE_OUT0 6 = CLK_M
7:0	0x0	SDMMC1_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.61 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC2_0

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

Bit	Reset	Description
31:29	CLK_M	SDMMC2_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 101 = pllE_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 5 = PLLE_OUT0 6 = CLK_M
7:0	0x0	SDMMC2_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.62 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC4_0

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

Bit	Reset	Description
31:29	CLK_M	SDMMC4_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 101 = pllE_out0, 110 = CLK_M 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 5 = PLLE_OUT0 6 = CLK_M
7:0	0x0	SDMMC4_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.63 CLK_RST_CONTROLLER_CLK_SOURCE_VFIR_0

Offset: 0x168 │ Read/Write: R/W │ Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	VFIR_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	VFIR_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.64 CLK_RST_CONTROLLER_CLK_SOURCE_HSI_0

Offset: 0x174 │ Read/Write: R/W │ Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	HSI_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	HSI_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.65 CLK_RST_CONTROLLER_CLK_SOURCE_UARTA_0

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

Bit	Reset	Description
31:29	CLK_M	UARTA_CLK_SRC: UART baud clock divisors by default come directly from the UART DLM/DLL registers in 16.0 format. Enable UART_DIV_ENB to use the 15.1 divisor below. 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M

Bit	Reset	Description
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 denotes 0.5x)

5.7.66 CLK_RST_CONTROLLER_CLK_SOURCE_UARTB_0

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

Bit	Reset	Description
31:29	CLK_M	UARTB_CLK_SRC: UART baud clock divisors by default come directly from the UART DLM/DLL registers in 16.0 format. Enable UART_DIV_ENB to use the 15.1 divisor below. 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
24	DISABLE	UARTB_DIV_ENB: 0=Use UART DLM/DLL, 1=Use divisor below 0 = DISABLE 1 = ENABLE
15:0	0x2	UARTB_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.67 CLK_RST_CONTROLLER_CLK_SOURCE_HOST1X_0

Offset: 0x180 │ Read/Write: R/W │ Reset: 0x00000000 (0b000xxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31:29	PLLM_OUT0	HOST1X_CLK_SRC: 000 = pllM_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = pllA_out0 0 = PLLM_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = PLLA_OUT0
15:8	0x0	HOST1X_IDLE_DIVISOR: N = Divide by (n+1) (lsb denotes 0.5x). If all 0s, this idle divisor field will not be used. For non-zero values, when host1x is idle, this field will be used instead of HOST1X_CLK_DIVISOR.
7:0	0x0	HOST1X_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.68 CLK_RST_CONTROLLER_CLK_SOURCE_HDMI_0

Offset: 0x18c │ Read/Write: R/W │ Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	HDMI_CLK_SRC: 000 = pllP_out0, 001 = pllM_out0, 010 = pllD_out0, 011 = pllA_out0, 100 = pllC_out0, 101 = pllD2_out0, 110 = clk_m, 101 = pllD2_out0, 111 = 1'b0. Non sequential mapping is used to preserve 2 MSBs to match legacy source select. 0 = PLLP_OUT0 1 = PLLM_OUT0 2 = PLLD_OUT0 3 = PLLA_OUT0 4 = PLLC_OUT0 5 = PLLD2_OUT0 6 = CLK_M
7:0	0x0	HDMI_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.69 CLK_RST_CONTROLLER_CLK_SOURCE_I2C2_0

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

Bit	Reset	Description
31:29	CLK_M	I2C2_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
15:0	0x0	I2C2_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 1.0x)

5.7.70 CLK_RST_CONTROLLER_CLK_SOURCE_EMC_0

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

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

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

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

As an alternative to writing one of the sequencing initiating fields, the FORCE_CC_TRIGGER is provided to force the state machine to run. This is not the recommended method.

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

- Initiates sequencing (value must change):
  - CLK_SOURCE_EMC.EMC_2X_CLK_SRC
  - EMC_2X_CLK_DIVISOR
  - MC_EMC_SAME_FREQ

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

Offset: 0x19c │ Read/Write: R/W │ Reset: 0x60000000 (0b0110000xxxxxxxx0xxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	EMC_2X_CLK_SRC: 000 = pllM_out0, 001 = pllC_out0, 010 = pllP_out0, 011 = clk_m, 100 = pllM_out_for_emc undivided PllM_out0, 101 = pllC2_out, 110 = pllC3_out, 111 = PLLC_UD. 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = CLK_M 4 = PLLM_UD 5 = PLLC2_OUT0 6 = PLLC3_OUT0 7 = PLLC_UD
28	0x0	CLK_DIS_SOC_MC: 1 = Disable the SOC leg of the mcclk (instead of setting emcclk = OSC, and killing OSC.
27	0x0	FORCE_CC_TRIGGER: 1 = force a trigger of clock change sequence even if EMC_2X_CLK_SRC/EMC_2X_CLK_DIVISOR fields are unchanged.
26	DISABLE	EMC_INVERT_DCD: 1 = Enable inversion of DCD (duty-cycle distortion), 0 = Disable inversion of DCD. 0 = DISABLE 1 = ENABLE See “Invert Duty-Cycle Distortion Control” in this section for more information.
25	0x0	USE_32KHZ_AS_CLK_M: 1 = CLK_M = 32kHz clock; 0 = CLK_M = OSC = clk_m
16	DISABLE	MC_EMC_SAME_FREQ: 0 = MC is 1/2 the frequency of EMC. 1 = MC has the same frequency as the EMC. 0 = DISABLE 1 = ENABLE
7:0	0x0	EMC_2X_CLK_DIVISOR: N = Divide by (n+1) (lsb denotes 0.5x)

5.7.71 CLK_RST_CONTROLLER_CLK_SOURCE_UARTC_0

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

Bit	Reset	Description
31:29	CLK_M	UARTC_CLK_SRC: UART baud clock divisors by default come directly from the UART DLM/DLL registers in 16.0 format. Enable UART_DIV_ENB to use the 15.1 divisor below. 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = CLK_M. 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M

Bit	Reset	Description
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 denotes 0.5x)

5.7.72 CLK_RST_CONTROLLER_CLK_SOURCE_VI_SENSOR_0

Offset: 0x1a8 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLM_OUT0	VI_SENSOR_CLK_SRC: 000 = pllM_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = pllA_out0 0 = PLLM_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = PLLA_OUT0
7:0	0x0	VI_SENSOR_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.73 CLK_RST_CONTROLLER_CLK_SOURCE_SPI4_0

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

Bit	Reset	Description
31:29	CLK_M	SPI4_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	SPI4_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.74 CLK_RST_CONTROLLER_CLK_SOURCE_I2C3_0

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

Bit	Reset	Description
31:29	CLK_M	I2C3_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
15:0	0x0	I2C3_CLK_DIVISOR: N = Divide by (n+1) (lsb denotes 1.0x)

5.7.75 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC3_0

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

Bit	Reset	Description
31:29	CLK_M	SDMMC3_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 101 = pllE_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 5 = PLLE_OUT0 6 = CLK_M
7:0	0x0	SDMMC3_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.76 CLK_RST_CONTROLLER_CLK_SOURCE_UARTD_0

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

Bit	Reset	Description
31:29	CLK_M	UARTD_CLK_SRC: UART baud clock divisors by default come directly from the UART DLM/DLL registers in 16.0 format. Enable UART_DIV_ENB to use the 15.1 divisor below. 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
24	DISABLE	UARTD_DIV_ENB: 0=Use UART DLM/DLL, 1=Use divisor below. 0 = DISABLE 1 = ENABLE
15:0	0X2	UARTD_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.77 CLK_RST_CONTROLLER_CLK_SOURCE_VDE_0

Offset: 0x1c8 │ Read/Write: R/W │ Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	VDE_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	VDE_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.78 CLK_RST_CONTROLLER_CLK_SOURCE_OWR_0

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

Bit	Reset	Description
31:29	CLK_M	OWR_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	OWR_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.79 CLK_RST_CONTROLLER_CLK_SOURCE_NOR_0

Offset: 0x1d0 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SNOR_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	SNOR_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.80 CLK_RST_CONTROLLER_CLK_SOURCE_CSITE_0

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

Bit	Reset	Description
31:29	CLK_M	CSITE_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	CSITE_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.81 CLK_RST_CONTROLLER_CLK_SOURCE_I2S0_0

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

Bit	Reset	Description
31:29	CLK_M	I2S0_CLK_SRC: 000 = pllA_out0, 010 = audio SYNC_CLK 1x or 2x, 100 = pllP_out0, 110 = clk_m 0 = PLLA_OUT0 2 = SYNC_CLK 4 = PLLP_OUT0 6 = CLK_M

Bit	Reset	Description
28	0x1	I2S0_MASTER_CLKEN: Reserved.
7:0	0x0	I2S0_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.82 CLK_RST_CONTROLLER_CLK_SOURCE_DTV_0

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

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

5.7.83 CLK_RST_CONTROLLER_CLK_SOURCE_MSENC_0

Offset: 0x1f0 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLM_OUT0	MSENC_CLK_SRC: 000 = pllM_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = pllA_out0 0 = PLLM_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = PLLA_OUT0
7:0	0x0	MSENC_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.84 CLK_RST_CONTROLLER_CLK_SOURCE_TSEC_0

Offset: 0x1f4 | Read/Write: R/W | Reset: 0x80000000 (0b100xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLM_OUT0	TSEC_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 101 = pllA_out0, 110 = CLK_M (osc) 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 5 = PLLA_OUT0 6 = CLK_M
7:0	0x0	TSEC_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.85 CLK_RST_CONTROLLER_CLK_SPARE2_0

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

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

5.7.86 CLK_RST_CONTROLLER_CLK_OUT_ENB_X_0

Offset: 0x280 | Read/Write: R/W | Reset: 0x01000000 (0bxxxxxx01x000x000x0xx0xxxx000xxx0)

Bit	Reset	Description
25	DISABLE	CLK_ENB_AMX1: Enable clock - AMX1

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
24	ENABLE	CLK_ENB_GPU: Enable clock - GPU 0 = DISABLE 1 = ENABLE
22	DISABLE	CLK_ENB_SOR0: Enable clock - SOR0 0 = DISABLE 1 = ENABLE
21	DISABLE	CLK_ENB_DPAUX: Enable clock - DPAUX 0 = DISABLE 1 = ENABLE
20	DISABLE	CLK_ENB_ADX1: Enable clock - ADX1 0 = DISABLE 1 = ENABLE
18	DISABLE	CLK_ENB_VIC: Enable clock - VIC 0 = DISABLE 1 = ENABLE
17	DISABLE	CLK_ENB_CLK72MHZ: Enable clock - CLK72MHZ 0 = DISABLE 1 = ENABLE
16	DISABLE	CLK_ENB_HDMI_AUDIO: Enable clock - HDMI AUDIO 0 = DISABLE 1 = ENABLE
14	DISABLE	CLK_ENB_EMC_DLL: Enable clock - EMC DLL 0 = DISABLE 1 = ENABLE
11	DISABLE	CLK_ENB_VIM2_CLK: Enable clock - CAMERA ifc 2 0 = DISABLE 1 = ENABLE
6	DISABLE	CLK_ENB_I2C6: Enable clock - I2C6 0 = DISABLE 1 = ENABLE
5	DISABLE	CLK_ENB_CAM_MCLK2: Enable clock - CAM_MCLK2 0 = DISABLE 1 = ENABLE
4	DISABLE	CLK_ENB_CAM_MCLK: Enable clock - CAM_MCLK 0 = DISABLE 1 = ENABLE
0	DISABLE	CLK_ENB_SPARE: Enable clock - SPARE 0 = DISABLE 1 = ENABLE

5.7.87 CLK_RST_CONTROLLER_CLK_ENB_X_SET_0

Offset: 0x284 │ Read/Write: R/W │ Reset: 0x01000000 (0bxxxxxx01x000x000x0xx0xxxx000xxx0)

Bit	Reset	Description
25	DISABLE	SET_CLK_ENB_AMX1: Set enable clock - AMX1 0 = DISABLE 1 = ENABLE
24	ENABLE	SET_CLK_ENB_GPU: Set enable clock - GPU 0 = DISABLE 1 = ENABLE
22	DISABLE	SET_CLK_ENB_SOR0: Set enable clock - SOR0 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
21	DISABLE	SET_CLK_ENB_DPAUX: Set enable clock - DPAUX 0 = DISABLE 1 = ENABLE
20	DISABLE	SET_CLK_ENB_ADX1: Set enable clock - ADX1 0 = DISABLE 1 = ENABLE
18	DISABLE	SET_CLK_ENB_VIC: Set enable clock - VIC 0 = DISABLE 1 = ENABLE
17	DISABLE	SET_CLK_ENB_CLK72MHZ: Set enable clock - CLK72MHZ 0 = DISABLE 1 = ENABLE
16	DISABLE	SET_CLK_ENB_HDMI_AUDIO: Set enable clock - HDMI AUDIO 0 = DISABLE 1 = ENABLE
14	DISABLE	SET_CLK_ENB_EMC_DLL: Set enable clock - EMC DLL 0 = DISABLE 1 = ENABLE
11	DISABLE	SET_CLK_ENB_VIM2_CLK: Set enable clock - CAMERA ifc 2 0 = DISABLE 1 = ENABLE
6	DISABLE	SET_CLK_ENB_I2C6: Set enable clock - I2C6 0 = DISABLE 1 = ENABLE
5	DISABLE	SET_CLK_ENB_CAM_MCLK2: Set enable clock - CAM_MCLK2 0 = DISABLE 1 = ENABLE
4	DISABLE	SET_CLK_ENB_CAM_MCLK: Set enable clock - CAM_MCLK 0 = DISABLE 1 = ENABLE
0	DISABLE	SET_CLK_ENB_SPARE: Set enable clock - SPARE 0 = DISABLE 1 = ENABLE

5.7.88 CLK_RST_CONTROLLER_CLK_ENB_X_CLR_0

Offset: 0x288 │ Read/Write: R/W │ Reset: 0x01000000 (0bxxxxxx01x000x000x0xx0xxxx000xxx0)

Bit	Reset	Description
25	DISABLE	CLR_CLK_ENB_AMX1: Clear enable clock - AMX1 0 = DISABLE 1 = ENABLE
24	ENABLE	CLR_CLK_ENB_GPU: Clear enable clock - GPU 0 = DISABLE 1 = ENABLE
22	DISABLE	CLR_CLK_ENB_SOR0: Clear enable clock - SOR0 0 = DISABLE 1 = ENABLE
21	DISABLE	CLR_CLK_ENB_DPAUX: Clear enable clock - DPAUX 0 = DISABLE 1 = ENABLE
20	DISABLE	CLR_CLK_ENB_ADX1: Clear enable clock - ADX1 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
18	DISABLE	CLR_CLK_ENB_VIC: Clear enable clock - VIC 0 = DISABLE 1 = ENABLE
17	DISABLE	CLR_CLK_ENB_CLK72MHZ: Clear enable clock - CLK72MHZ 0 = DISABLE 1 = ENABLE
16	DISABLE	CLR_CLK_ENB_HDMI_AUDIO: Clear enable clock - HDMI AUDIO 0 = DISABLE 1 = ENABLE
14	DISABLE	CLR_CLK_ENB_EMC_DLL: Clear enable clock - EMC DLL 0 = DISABLE 1 = ENABLE
11	DISABLE	CLR_CLK_ENB_VIM2_CLK: Clear enable clock - CAMERA ifc 2 0 = DISABLE 1 = ENABLE
6	DISABLE	CLR_CLK_ENB_I2C6: Clear enable clock - I2C6 0 = DISABLE 1 = ENABLE
5	DISABLE	CLR_CLK_ENB_CAM_MCLK2: Clear enable clock - CAM_MCLK2 0 = DISABLE 1 = ENABLE
4	DISABLE	CLR_CLK_ENB_CAM_MCLK: Clear enable clock - CAM_MCLK 0 = DISABLE 1 = ENABLE
0	DISABLE	CLR_CLK_ENB_SPARE: Clear enable clock - SPARE 0 = DISABLE 1 = ENABLE

5.7.89 CLK_RST_CONTROLLER_RST_DEVICES_X_0

Offset: 0x28c │ Read/Write: R/W │ Reset: 0xff740041 (0b11111111x111x1xxxxxxxxxxx1xxxxx1)

Bit	Reset	Description
31	ENABLE	SWR_AFC0_RST: Reset - AFC0 0 = DISABLE 1 = ENABLE
30	ENABLE	SWR_AFC1_RST: Reset – AFC1 0 = DISABLE 1 = ENABLE
29	ENABLE	SWR_AFC2_RST: Reset – AFC2 0 = DISABLE 1 = ENABLE
28	ENABLE	SWR_AFC3_RST: Reset – AFC3 0 = DISABLE 1 = ENABLE
27	ENABLE	SWR_AFC4_RST: Reset – AFC4 0 = DISABLE 1 = ENABLE
26	ENABLE	SWR_AFC5_RST: Reset – AFC5 0 = DISABLE 1 = ENABLE
25	ENABLE	SWR_AMX1_RST: Reset – AMX1 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
24	ENABLE	SWR_GPU_RST: Reset - GPU 0 = DISABLE 1 = ENABLE
22	ENABLE	SWR_SOR0_RST: Reset – SOR0 0 = DISABLE 1 = ENABLE
21	ENABLE	SWR_DPAUX_RST: Reset - DPAUX 0 = DISABLE 1 = ENABLE
20	ENABLE	SWR_ADX1_RST: Reset – ADX1 0 = DISABLE 1 = ENABLE
18	ENABLE	SWR_VIC_RST: Reset - VIC 0 = DISABLE 1 = ENABLE
6	ENABLE	SWR_I2C6_RST: Reset – I2C6 0 = DISABLE 1 = ENABLE
0	ENABLE	SWR_SPARE_RST: Reset - SPARE 0 = DISABLE 1 = ENABLE

5.7.90 CLK_RST_CONTROLLER_RST_DEV_X_SET_0

Offset: 0x290 │ Read/Write: R/W │ Reset: 0xff740041 (0b11111111x111x1xxxxxxxxxxx1xxxxx1)

Bit	Reset	Description
31	ENABLE	SET_AFC0_RST: Set reset - AFC0 0 = DISABLE 1 = ENABLE
30	ENABLE	SET_AFC1_RST: Set reset – AFC1 0 = DISABLE 1 = ENABLE
29	ENABLE	SET_AFC2_RST: Set reset – AFC2 0 = DISABLE 1 = ENABLE
28	ENABLE	SET_AFC3_RST: Set reset – AFC3 0 = DISABLE 1 = ENABLE
27	ENABLE	SET_AFC4_RST: Set reset – AFC4 0 = DISABLE 1 = ENABLE
26	ENABLE	SET_AFC5_RST: Set reset – AFC5 0 = DISABLE 1 = ENABLE
25	ENABLE	SET_AMX1_RST: Set reset – AMX1 0 = DISABLE 1 = ENABLE
24	ENABLE	SET_GPU_RST: Set reset - GPU 0 = DISABLE 1 = ENABLE
22	ENABLE	SET_SOR0_RST: Set reset – SOR0 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
21	ENABLE	SET_DPAUX_RST: Set reset - DPAUX 0 = DISABLE 1 = ENABLE
20	ENABLE	SET_ADX1_RST: Set reset – ADX1 0 = DISABLE 1 = ENABLE
18	ENABLE	SET_VIC_RST: Set reset - VIC 0 = DISABLE 1 = ENABLE
6	ENABLE	SET_I2C6_RST: Set reset – I2C6 0 = DISABLE 1 = ENABLE
0	0x1	SET_SPARE_RST: Set reset - SPARE 0 = DISABLE 1 = ENABLE

5.7.91 CLK_RST_CONTROLLER_RST_DEV_X_CLR_0

Offset: 0x294 │ Read/Write: R/W │ Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
31	ENABLE	CLR_AFC0_RST: Clear reset - AFC0 0 = DISABLE 1 = ENABLE
30	ENABLE	CLR_AFC1_RST: Clear reset – AFC1 0 = DISABLE 1 = ENABLE
29	ENABLE	CLR_AFC2_RST: Clear reset – AFC2 0 = DISABLE 1 = ENABLE
28	ENABLE	CLR_AFC3_RST: Clear reset – AFC3 0 = DISABLE 1 = ENABLE
27	ENABLE	CLR_AFC4_RST: Clear reset – AFC4 0 = DISABLE 1 = ENABLE
26	ENABLE	CLR_AFC5_RST: Clear reset – AFC5 0 = DISABLE 1 = ENABLE
25	ENABLE	CLR_AMX1_RST: Clear reset – AMX1 0 = DISABLE 1 = ENABLE
24	ENABLE	CLR_GPU_RST: Clear reset - GPU 0 = DISABLE 1 = ENABLE
22	ENABLE	CLR_SOR0_RST: Clear reset – SOR0 0 = DISABLE 1 = ENABLE
21	ENABLE	CLR_DPAUX_RST: Clear reset - DPAUX 0 = DISABLE 1 = ENABLE
20	ENABLE	CLR_ADX1_RST: Clear reset – ADX1 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
18	ENABLE	CLR_VIC_RST: Clear reset - VIC 0 = DISABLE 1 = ENABLE
6	ENABLE	CLR_I2C6_RST: Clear reset – I2C6 0 = DISABLE 1 = ENABLE
0	0x1	CLR_SPARE_RST: Clear reset - SPARE 0 = DISABLE 1 = ENABLE

5.7.92 CLK_RST_CONTROLLER_DFLL_BASE_0

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

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

5.7.93 CLK_RST_CONTROLLER_RST_DEV_L_SET_0

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

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

Offset: 0x300 │ Read/Write: R/W │ Reset: 0x7cd7dXXX (0b011111xx11x1x11111x1111x11xx100x)

Bit	R/W	Reset	Description
31	RW	0x0	SET_CACHE2_RST: Set reset for the COP cache controller. 0 = DISABLE 1 = ENABLE
30	RW	0x1	SET_I2S0_RST: Set reset for the I2S0 controller. 0 = DISABLE 1 = ENABLE
29	RW	0x1	SET_VCP_RST: Set reset for the vector coprocessor. 0 = DISABLE 1 = ENABLE
28	RW	0x1	SET_HOST1X_RST: Set reset for the HOST1X. 0 = DISABLE 1 = ENABLE
27	RW	0x1	SET_DISP1_RST: Set reset for the DISP1 controller. 0 = DISABLE 1 = ENABLE
26	RW	0x1	SET_DISP2_RST: Set reset for the DISP2 controller. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
23	RW	0x1	SET_ISP_RST: Set reset for the ISP controller. 0 = DISABLE 1 = ENABLE
22	RW	0x1	SET_USBD_RST: Set reset for the USB controller. 0 = DISABLE 1 = ENABLE
20	RW	0x1	SET_VI_RST: Set reset for the VI controller. 0 = DISABLE 1 = ENABLE
18	RW	0x1	SET_I2S2_RST: Set reset for the I2S2 controller. 0 = DISABLE 1 = ENABLE
17	RW	0x1	SET_PWM_RST: Set reset for the Pulse Width Modulator. 0 = DISABLE 1 = ENABLE
15	RW	0x1	SET_SDMMC4_RST: Set reset for the SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	RW	0x1	SET_SDMMC1_RST: Set reset for the SDMMC1 controller. 0 = DISABLE 1 = ENABLE
12	RW	0x1	SET_I2C1_RST: Set reset for the I2C1 controller. 0 = DISABLE 1 = ENABLE
11	RW	0x1	SET_I2S1_RST: Set reset for the I2S1 controller. 0 = DISABLE 1 = ENABLE
10	RW	0x1	SET_SPDIF_RST: Set reset for the S/PDIF controller. 0 = DISABLE 1 = ENABLE
9	RW	0x1	SET_SDMMC2_RST: Set reset for the SDMMC2 controller. 0 = DISABLE 1 = ENABLE
8	RW	0x0	SET_GPIO_RST: Set reset for the GPIO controller. 0 = DISABLE 1 = ENABLE
7	RW	0x1	SET_UARTB_RST: Set reset for the UARTB/VFIR controller. 0 = DISABLE 1 = ENABLE
6	RW	0x1	SET_UARTA_RST: Set reset for the UARTA controller. 0 = DISABLE 1 = ENABLE
5	RO	x	SET_TMR_RST: Reserved
3	RW	0x1	SET_ISPB_RST: Set reset - ISPB
2	RW	0x0	SET_TRIG_SYS_RST: Write 1 to pulse the System Reset signal. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
1	RW	0x0	SET_COP_RST: Set/reset COP. 0 = DISABLE 1 = ENABLE
0	RO	x	SET_CPU_RST: Set the reset for the CPU. [CCLK Multi-Address]. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE

5.7.94 CLK_RST_CONTROLLER_RST_DEV_L_CLR_0

Offset: 0x304 | Read/Write: R/W | Reset: 0x7cd7dXXX (0b011111xx11x1x11111x1111x11xx1x0x)

Bit	R/W	Reset	Description
31	RW	0x0	CLR_CACHE2_RST: Clear reset for the COP cache controller. 0 = DISABLE 1 = ENABLE
30	RW	0x1	CLR_I2S0_RST: Clear reset for the I2S0 controller. 0 = DISABLE 1 = ENABLE
29	RW	0x1	CLR_VCP_RST: Clear reset for the vector coprocessor. 0 = DISABLE 1 = ENABLE
28	RW	0x1	CLR_HOST1X_RST: Clear reset for the HOST1X. 0 = DISABLE 1 = ENABLE
27	RW	0x1	CLR_DISP1_RST: Clear reset for the DISP1 controller. 0 = DISABLE 1 = ENABLE
26	RW	0x1	CLR_DISP2_RST: Clear reset for the DISP2 controller. 0 = DISABLE 1 = ENABLE.
23	RW	0x1	CLR_ISP_RST: Clear reset for the ISP controller. 0 = DISABLE 1 = ENABLE
22	RW	0x1	CLR_USBD_RST: Clear reset for the USB controller. 0 = DISABLE 1 = ENABLE
20	RW	0x1	CLR_VI_RST: Clear reset for the VI controller. 0 = DISABLE 1 = ENABLE
18	RW	0x1	CLR_I2S2_RST: Clear reset for the I2S 2 controller. 0 = DISABLE 1 = ENABLE
17	RW	0x1	CLR_PWM_RST: Clear reset for the Pulse Width Modulator. 0 = DISABLE 1 = ENABLE
15	RW	0x1	CLR_SDMMC4_RST: Clear reset for the SDMMC4 controller. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
14	RW	0x1	CLR_SDMMC1_RST: Clear reset for the SDMMC1 controller. 0 = DISABLE 1 = ENABLE
12	RW	0x1	CLR_I2C1_RST: Clear reset for the I2C1 controller. 0 = DISABLE 1 = ENABLE
11	RW	0x1	CLR_I2S1_RST: Clear reset for the I2S1 controller. 0 = DISABLE 1 = ENABLE
10	RW	0x1	CLR_SPDIF_RST: Clear reset for the S/PDIF controller. 0 = DISABLE 1 = ENABLE
9	RW	0x1	CLR_SDMMC2_RST: Clear reset for the SDMMC2 controller. 0 = DISABLE 1 = ENABLE
8	RO	X	CLR_GPIO_RST: Clear reset for the GPIO controller. 0 = DISABLE 1 = ENABLE
7	RW	0x1	CLR_UARTB_RST: Clear reset for the UARTB/VFIR controller. 0 = DISABLE 1 = ENABLE
6	RW	0x1	CLR_UARTA_RST: Clear reset for the UARTA controller. 0 = DISABLE 1 = ENABLE
5	RO	X	CLR_TMR_RST: Reserved
3	RW	0x1	CLR_ISPB_RST: Clear reset - ISBP 0 = DISABLE 1 = ENABLE
1	RW	0x0	CLR_COP_RST: Clear reset for the COP. 0 = DISABLE 1 = ENABLE
0	RW	X	CLR_CPU_RST: Clear reset for the CPU. [CCLK Multi-Address] This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE

5.7.95 CLK_RST_CONTROLLER_RST_DEV_H_SET_0

Offset: 0x308 │ Read/Write: R/W │ Reset: 0xefddf32X (0b111x111111x111x11111x0110x1xx11x)

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

Bit	R/W	Reset	Description
29	RW	0x1	SET_VDE_RST: Set reset for the VDE controller. 0 = DISABLE 1 = ENABLE
27	RW	0x1	SET_USB3_RST: Set reset for the USB3 controller. 0 = DISABLE 1 = ENABLE
26	RW	0x1	SET_USB2_RST: Set reset for the USB2 controller. 0 = DISABLE 1 = ENABLE
25	RW	0x1	SET_EMC_RST: Set reset for the EMC controller. 0 = DISABLE 1 = ENABLE
24	RW	0x1	SET_MIPI_CAL_RST: Set reset for the MIPI CAL logic. 0 = DISABLE 1 = ENABLE
23	RW	0x1	SET_UARTC_RST: Set reset for the UARTC controller. 0 = DISABLE 1 = ENABLE
22	RW	0x1	SET_I2C2_RST: Set reset for the I2C2 controller. 0 = DISABLE 1 = ENABLE
20	RW	0x1	SET_CSI_RST: Set reset for the CSI controller. 0 = DISABLE 1 = ENABLE
19	RW	0x1	SET_HDMI_RST: Set reset for the HDMI 0 = DISABLE 1 = ENABLE
18	RW	0x1	SET_HSI_RST: Set reset for the MIPI baseband HSI controller. 0 = DISABLE 1 = ENABLE
16	RW	0x1	SET_DSI_RST: Set reset for the DSI controller. 0 = DISABLE 1 = ENABLE
15	RW	0x1	SET_ I2C5_RST: Set reset for the I2C5 controller. 0 = DISABLE 1 = ENABLE
14	RW	0x1	SET_SPI3_RST: Set reset for the SPI3 controller. 0 = DISABLE 1 = ENABLE
12	RW	0x1	SET_SPI2_RST: Set reset for the SPI2 controller. 0 = DISABLE 1 = ENABLE
10	RW	0x0	SET_SNOR_RST: Set reset for the NOR Flash controller. 0 = DISABLE 1 = ENABLE
9	RW	0x1	SET_SPI1_RST: Set reset for the SPI1 Controller. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
8	RW	0x1	SET_KFUSE_RST: Set reset for the KFuse controller. 0 = DISABLE 1 = ENABLE
5	RW	0x1	SET_STAT_MON_RST: Set reset for the statistic monitor 0 = DISABLE 1 = ENABLE
2	RW	0x1	SET_APBDMA_RST: Set reset for the APB-DMA. 0 = DISABLE 1 = ENABLE
1	RW	0x1	SET_AHBDMA_RST: Set reset for the AHB-DMA. 0 = DISABLE 1 = ENABLE
0	RO	X	SET_MEM_RST: Set reset MC. This bit is disabled. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE

5.7.96 CLK_RST_CONTROLLER_RST_DEV_H_CLR_0

Offset: 0x30c │ Read/Write: R/W │ Reset: 0xefddf32X (0b111x111111x111x11111x0110x1xx11x)

Bit	R/W	Reset	Description
31	RW	0x1	CLR_BSEV_RST: Clear reset for the BSEV controller. 0 = DISABLE 1 = ENABLE
30	RW	0x1	CLR_BSEA_RST: Clear reset for the BSEA controller. 0 = DISABLE 1 = ENABLE
29	RW	0x1	CLR_VDE_RST: Clear reset for the VDE controller. 0 = DISABLE 1 = ENABLE
27	RW	0x1	CLR_USB3_RST: Clear reset for the USB3 controller. 0 = DISABLE 1 = ENABLE
26	RW	0x1	CLR_USB2_RST: Clear reset for the USB2 controller. 0 = DISABLE 1 = ENABLE
25	RW	0x1	CLR_EMC_RST: Clear reset for the EMC controller. 0 = DISABLE 1 = ENABLE
24	RW	0X1	CLR_MIPI_CAL_RST: Clear reset for the MIPI CAL logic. 0 = DISABLE 1 = ENABLE
23	RW	0x1	CLR_UARTC_RST: Clear reset for the UARTC controller. 0 = DISABLE 1 = ENABLE
22	RW	0x1	CLR_I2C2_RST: Clear reset for the I2C2 controller. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
20	RW	0x1	CLR_CSI_RST: Clear reset for the CSI controller. 0 = DISABLE 1 = ENABLE
19	RW	0x1	CLR_HDMI_RST: Clear reset for the HDMI. 0 = DISABLE 1 = ENABLE
18	RW	0x1	CLR_MIPI_RST: Clear reset for the MIPI baseband HSI controller. 0 = DISABLE 1 = ENABLE
16	RW	0x1	CLR_DSI_RST: Clear reset for the DSI controller. 0 = DISABLE 1 = ENABLE
15	RW	0x1	CLR_ I2C5_RST: Clear reset for the I2C5 controller. 0 = DISABLE 1 = ENABLE
14	RW	0x1	CLR_SPI3_RST: Clear reset for the SPI3 controller. 0 = DISABLE 1 = ENABLE
12	RW	0x1	CLR_SPI2_RST: Clear reset for the SPI2 controller. 0 = DISABLE 1 = ENABLE
10	RW	0x0	CLR_SNOR_RST: Clear reset for the NOR Flash controller. 0 = DISABLE 1 = ENABLE
9	RW	0x1	CLR_SPI1_RST: Clear reset for the SPI1 controller. 0 = DISABLE 1 = ENABLE
8	RW	0x1	CLR_KFUSE_RST: Clear reset for the KFuse controller. 0 = DISABLE 1 = ENABLE
5	RW	0x1	CLR_STAT_MON_RST: Clear reset for the statistic monitor. 0 = DISABLE 1 = ENABLE
2	RW	0x1	CLR_APBDMA_RST: Clear reset for the APB-DMA. 0 = DISABLE 1 = ENABLE
1	RW	0x1	CLR_AHBDMA_RST: Clear reset for the AHB-DMA. 0 = DISABLE 1 = ENABLE
0	RO	X	CLR_MEM_RST: Clear reset for the MC. This bit is disabled. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE

5.7.97 CLK_RST_CONTROLLER_RST_DEV_U_SET_0

Offset: 0x310 | Read/Write: R/W | Reset: 0x8a8ed5fe (0b1xxx1x1x1xxx111x110101011111111x)

Bit	Reset	Description
31	0x1	SET_XUSB_DEV_RST: Set reset for the XUSB DEV logic. 0 = DISABLE 1 = ENABLE
27	0x1	SET_MSENC_RST: Set reset for the MSENC logic. 0 = DISABLE 1 = ENABLE
25	0x1	SET_XUSB_HOST_RST: Set reset for the XUSB HOST logic. 0 = DISABLE 1 = ENABLE
23	0x1	SET_EMUCIF_RST: Set reset for the EMUCIF logic 0 = DISABLE 1 = ENABLE
19	0x1	SET_TSEC_RST: Set reset for the TSEC logic 0 = DISABLE 1 = ENABLE
18	0x1	SET_DSIB_RST: Set reset for the DSIB controller 0 = DISABLE 1 = ENABLE
17	0x1	SET_I2C_SLOW_RST: Set reset for the I2C_SLOW 0 = DISABLE 1 = ENABLE
15	0x1	SET_DTV_RST: Set reset for the DTV controller. 0 = DISABLE 1 = ENABLE
14	0x1	SET_SOC_THERM_RST: Set reset for the SOC_THERM controller. 0 = DISABLE 1 = ENABLE
13	0x0	SET_TRACECLKIN_RST: Set reset for the TRACECLKIN controller. 0 = DISABLE 1 = ENABLE
11	0x0	SET_AVPUCQ_RST: Set reset for the AVPUCQ logic. 0 = DISABLE 1 = ENABLE
10	0x1	SET_PCIEXCLK_RST: Set reset for the PCIEXCLK logic. 0 = DISABLE 1 = ENABLE
9	0x0	SET_CSITE_RST: Set reset for the CSITE controller. 0 = DISABLE 1 = ENABLE
8	0x1	SET_AFI_RST: Set reset for the AFI controller. 0 = DISABLE 1 = ENABLE
7	0x1	SET_OWR_RST: Set reset for the OWR controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	0x1	SET_PCIE_RST: Set reset for the PCIe controller. 0 = DISABLE 1 = ENABLE
5	0x1	SET_SDMMC3_RST: Set reset for the SDMMC3 controller. 0 = DISABLE 1 = ENABLE
4	0x1	SET_SPI4_RST: Set reset for the SPI4 controller. 0 = DISABLE 1 = ENABLE
3	0x1	SET_I2C3_RST: Set reset for the I2C3 controller. 0 = DISABLE 1 = ENABLE
2	0x1	SET_UARTE_RST: Reserved.
1	0x1	SET_UARTD_RST: Set reset for the UARTD controller. 0 = DISABLE 1 = ENABLE

5.7.98 CLK_RST_CONTROLLER_RST_DEV_U_CLR_0

Offset: 0x314 │ Read/Write: R/W │ Reset: 0x8a8ed5fe (0b1xxx1x1x1xxx111x110101011111111x)

Bit	Reset	Description
31	0x1	CLR_XUSB_DEV_RST: Clear reset for the XUSB DEV logic. 0 = DISABLE 1 = ENABLE
27	0x1	CLR_MSENC_RST: Clear reset for the MSENC logic 0 = DISABLE 1 = ENABLE
25	0x1	CLR_XUSB_HOST_RST: Clear reset for the XUSB HOST logic. 0 = DISABLE 1 = ENABLE
23	0x1	CLR_EMUCIF_RST: Clear reset for the EMUCIF logic 0 = DISABLE 1 = ENABLE
19	0x1	CLR_TSEC_RST: Clear reset for the TSEC logic 0 = DISABLE 1 = ENABLE
18	0x1	CLR_DSIB_RST: Clear reset for the DSIB Controller 0 = DISABLE 1 = ENABLE
17	0x1	CLR_I2C_SLOW_RST: Clear reset for the I2C_SLOW 0 = DISABLE 1 = ENABLE
15	0x1	CLR_DTV_RST: Clear reset for the DTV controller 0 = DISABLE 1 = ENABLE
14	0x1	CLR_SOC_THERM_RST: Clear reset for the SOC_THERM controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
13	0x0	CLR_TRACECLKIN_RST: Clear reset for the TRACECLKIN controller. 0 = DISABLE 1 = ENABLE
11	0x0	CLR_AVPUCQ_RST: Clear reset for the AVPUCQ logic. 0 = DISABLE 1 = ENABLE
10	0x1	CLR_PCIEXCLK_RST: Clear reset for the PCIEXCLK logic. 0 = DISABLE 1 = ENABLE
9	0x0	CLR_CSITE_RST: Clear reset for the CSITE controller. 0 = DISABLE 1 = ENABLE
8	0x1	CLR_AFI_RST: Clear reset for the AFI controller. 0 = DISABLE 1 = ENABLE
7	0x1	CLR_OWR_RST: Clear reset for the OWR controller. 0 = DISABLE 1 = ENABLE
6	0x1	CLR_PCIE_RST: Clear reset for the PCIe controller. 0 = DISABLE 1 = ENABLE
5	0x1	CLR_SDMMC3_RST: Clear reset for the SDMMC3 controller. 0 = DISABLE 1 = ENABLE
4	0x1	CLR_SPI4_RST: Clear reset for the SPI4 controller. 0 = DISABLE 1 = ENABLE
3	0x1	CLR_I2C3_RST: Clear reset for the I2C3 controller. 0 = DISABLE 1 = ENABLE
2	0x1	CLR_UARTE_RST: Reserved.
1	0x1	CLR_UARTD_RST: Clear reset for the UARTD controller. 0 = DISABLE 1 = ENABLE

5.7.99 CLK_RST_CONTROLLER_CLK_ENB_L_SET_0

Offset: 0x320 │ Read/Write: R/W │ Reset: 0x80000130 (0b100000xx00x0x00000x0000100110xx0)

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

Bit	Reset	Description
29	0x0	SET_CLK_ENB_VCP: Set enable clock to vector coprocessor. 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
23	0x0	SET_CLK_ENB_ISP: Set enable clock to ISP controller. 0 = DISABLE 1 = ENABLE
22	0x0	SET_CLK_ENB_USBD: Set enable clock to USB controller 0 = DISABLE 1 = ENABLE
20	0x0	SET_CLK_ENB_VI: Set enable clock to VI controller. 0 = DISABLE 1 = ENABLE
18	0x0	SET_CLK_ENB_I2S2: Set enable clock to I2S2 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
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 S/PDIF 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

Bit	Reset	Description
7	0x0	SET_CLK_ENB_UARTB: Set enable clock to UARTB/VFIR controller 0 = DISABLE 1 = ENABLE
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
3	0x0	SET_CLK_ENB_ISPB: Set enable clock - ISPB 0 = DISABLE 1 = ENABLE
0	0x0	SET_CLK_ENB_CPU: Set enable clock to CPU. [CCLK Multi-Address] 0 = DISABLE 1 = ENABLE

5.7.100 CLK_RST_CONTROLLER_CLK_ENB_L_CLR_0

Offset: 0x324 │ Read/Write: R/W │ Reset: 0x80000130 (0b100000xx00x0x00000x0000100110xx0)

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 coprocessor. 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
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

Bit	Reset	Description
20	0x0	CLR_CLK_ENB_VI: Clear enable clock to VI controller. 0 = DISABLE 1 = ENABLE
18	0x0	CLR_CLK_ENB_I2S2: Clear enable clock to I2S2 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
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 S/PDIF 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
4	0x1	CLR_CLK_ENB_RTC: Clear enable clock to RTC controller 0 = DISABLE 1 = ENABLE
3	0x0	CLR_CLK_ENB_ISPB: Clear enable clock - ISPB 0 = DISABLE 1 = ENABLE
0	0x0	CLR_CLK_ENB_CPU: Clear enable clock to CPU. 0 = DISABLE 1 = ENABLE

5.7.101 CLK_RST_CONTROLLER_CLK_ENB_H_SET_0

Offset: 0x328 | Read/Write: R/W | Reset: 0x00000480 (0b000x000000x000x00000x1001000x000)

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
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
24	0x0	SET_CLK_ENB_MIPI_CAL: Set enable clock to MIPI CAL logic. 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
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
18	0x0	SET_CLK_ENB_HSI: Set enable clock to MIPI baseband HSI 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_ I2C5: Set enable clock to I2C5 controller. 0 = DISABLE 1 = ENABLE
14	0x0	SET_CLK_ENB_SPI3: Set enable clock to SPI 3 controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
12	0x0	SET_CLK_ENB_SPI2: Set enable clock to 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_SPI1: Set enable clock to SPI1 controller. 0 = DISABLE 1 = ENABLE
8	0x0	SET_CLK_ENB_KFUSE: Set enable clock to KFUSE controller. 0 = DISABLE 1 = ENABLE
7	0x1	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
0	0x0	SET_CLK_ENB_MEM: Set enable clock to MC. 0 = DISABLE 1 = ENABLE

5.7.102 CLK_RST_CONTROLLER_CLK_ENB_H_CLR_0

Offset: 0x32c │ Read/Write: R/W │ Reset: 0x00000480 (0b000x000000x000x00000x1001000x000)

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

Bit	Reset	Description
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
24	0x0	CLR_CLK_ENB_MIPI_CAL: Clear enable clock to MIPI CAL logic. 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
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_HSI: Clear enable clock to MIPI base-band HSI 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_ I2C5: Clear enable clock to -I2C5 controller. 0 = DISABLE 1 = ENABLE
14	0x0	CLR_CLK_ENB_SPI3: Clear enable clock to SPI3 controller. 0 = DISABLE 1 = ENABLE
12	0x0	CLR_CLK_ENB_SPI2: Clear enable clock to SPI2 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_SPI1: Clear enable clock to SPI1 Controller. 0 = DISABLE 1 = ENABLE
8	0x0	CLR_CLK_ENB_KFUSE: Clear enable clock to KFUSE controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	0x1	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

5.7.103 CLK_RST_CONTROLLER_CLK_ENB_U_SET_0

Offset: 0x330 │ Read/Write: R/W │ Reset: 0x01f82a00 (0b00000x011111100x00101x1000000x0x)

Bit	Reset	Description
31	0x0	SET_CLK_ENB_XUSB_DEV: Set enable clock to XUSB DEV. 0 = DISABLE 1 = ENABLE
30	0x0	SET_CLK_ENB_DEV1_OUT: Set enable clock to DEV1 pad. 0 = DISABLE 1 = ENABLE
29	0x0	SET_CLK_ENB_DEV2_OUT: Set enable clock to DEV2 pad. 0 = DISABLE 1 = ENABLE
28	0x0	SET_CLK_ENB_SUS_OUT: Set enable clock to SUS pad. 0 = DISABLE 1 = ENABLE
27	0x1	SET_CLK_ENB_MSENC: Set enable clock to the MSENC clock. 0 = DISABLE 1 = ENABLE
25	0x0	SET_CLK_ENB_XUSB_HOST: Set enable clock to XUSB HOST. 0 = DISABLE 1 = ENABLE
24	0x1	SET_CLK_ENB_CRAM2: Set enable clock for the COP cache RAM clock. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
23	0x1	SET_CLK_ENB_IRAMD: Set enable clock for the IRAMD clock. 0 = DISABLE 1 = ENABLE
22	0x1	SET_CLK_ENB_IRAMC: Set enable clock for the IRAMC clock. 0 = DISABLE 1 = ENABLE
21	0x1	SET_CLK_ENB_IRAMB: Set enable clock for the IRAMB clock. 0 = DISABLE 1 = ENABLE
20	0x1	SET_CLK_ENB_IRAMA: Set enable clock for the IRAMB clock. 0 = DISABLE 1 = ENABLE
19	0x1	SET_CLK_ENB_TSEC: Set enable clock for the TSEC clock. 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
15	0x0	SET_CLK_ENB_DTV: Set enable clock to DTV Controller. Moved to U:15 0 = DISABLE 1 = ENABLE
14	0x0	SET_CLK_ENB_SOC_THERM: Set enable clock to SOC_THERM controller. 0 = DISABLE 1 = ENABLE
13	0x1	SET_CLK_ENB_TRACECLKIN: Set enable clock to TRACECLKIN. 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
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

Bit	Reset	Description
4	0x0	SET_CLK_ENB_SPI4: Set enable clock to SPI4. 0 = DISABLE 1 = ENABLE
3	0x0	SET_CLK_ENB_I2C3: Set enable clock to I2C3. 0 = DISABLE 1 = ENABLE
1	0x0	SET_CLK_ENB_UARTD: Set enable clock to UARTD. 0 = DISABLE 1 = ENABLE

5.7.104 CLK_RST_CONTROLLER_CLK_ENB_U_CLR_0

Offset: 0x334 │ Read/Write: R/W │ Reset: 0x01f02a00 (0b00000x011111000x00101x1000000x0x)

Bit	Reset	Description
31	0x0	CLR_CLK_ENB_XUSB_DEV: Clear enable clock to XUSB DEV. 0 = DISABLE 1 = ENABLE
30	0x0	CLR_CLK_ENB_DEV1_OUT: Clear enable clock to DEV1 pad. 0 = DISABLE 1 = ENABLE
29	0x0	CLR_CLK_ENB_DEV2_OUT: Clear enable clock to DEV2 pad. 0 = DISABLE 1 = ENABLE
28	0x0	CLR_CLK_ENB_SUS_OUT: Clear enable clock to SUS pad. 0 = DISABLE 1 = ENABLE
27	0x0	CLR_CLK_ENB_MSENC: Clear enable clock to the MSENC clock. 0 = DISABLE 1 = ENABLE
25	0x0	CLR_CLK_ENB_XUSB_HOST: Clear enable clock to XUSB HOST. 0 = DISABLE 1 = ENABLE
24	0x1	CLR_CLK_ENB_CRAM2: Clear enable clock to the COP cache RAM clock. 0 = DISABLE 1 = ENABLE
23	0x1	CLR_CLK_ENB_IRAMD: Clear enable clock to the IRAMD clock. 0 = DISABLE 1 = ENABLE
22	0x1	CLR_CLK_ENB_IRAMC: Clear enable clock to the IRAMC clock. 0 = DISABLE 1 = ENABLE
21	0x1	CLR_CLK_ENB_IRAMB: Clear enable clock to the IRAMB clock. 0 = DISABLE 1 = ENABLE
20	0x1	CLR_CLK_ENB_IRAMA: Clear enable clock to the IRAMB clock. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
19	0x0	CLR_CLK_ENB_TSEC: Clear enable clock to the TSEC clock. 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
15	0x0	CLR_CLK_ENB_DTV: Clear enable clock to DTV controller. 0 = DISABLE 1 = ENABLE
14	0x0	CLR_CLK_ENB_SOC_THERM: Clear enable clock to SOC_THERM controller. 0 = DISABLE 1 = ENABLE
13	0x1	CLR_CLK_ENB_TRACECLKIN: Clear enable clock to TRACECLKIN. 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_SPI4: Clear enable clock to SPI4. 0 = DISABLE 1 = ENABLE
3	0x0	CLR_CLK_ENB_I2C3: Clear enable clock to I2C3. 0 = DISABLE 1 = ENABLE
1	0x0	CLR_CLK_ENB_UARTD: Clear enable clock to UARTD. 0 = DISABLE 1 = ENABLE

5.7.105 CLK_RST_CONTROLLER_CCPLEX_PG_SM_OVRD_0

CCPLEX Power Gate State-Machine Overrides

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

Offset: 0x33c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23	0x0	CLKSTOP_OVRD_SCPU_NC: 1 = Clamp the corresponding clocks 0 = DISABLE 1 = ENABLE
22	0x0	CLKSTOP_OVRD_SCPU_0: 1 = Clamp the corresponding clocks 0 = DISABLE 1 = ENABLE
21	0x0	CLKSTOP_OVRD_FCPU_RAIL: 1 = Clamp the corresponding clocks 0 = DISABLE 1 = ENABLE
20	0x0	CLKSTOP_OVRD_FCPU_NC: 1 = Clamp the corresponding clocks 0 = DISABLE 1 = ENABLE
19	0x0	CLKSTOP_OVRD_FCPU_3: 1 = Clamp the corresponding clocks 0 = DISABLE 1 = ENABLE
18	0x0	CLKSTOP_OVRD_FCPU_2: 1 = Clamp the corresponding clocks 0 = DISABLE 1 = ENABLE
17	0x0	CLKSTOP_OVRD_FCPU_1: 1 = Clamp the corresponding clocks 0 = DISABLE 1 = ENABLE
16	0x0	CLKSTOP_OVRD_FCPU_0: 1 = Clamp the corresponding clocks 0 = DISABLE 1 = ENABLE
15	0x0	RST_OVRD_SCPU_NC: 1 = Assert the corresponding reset 0 = DISABLE 1 = ENABLE
14	0x0	RST_OVRD_SCPU_0: 1 = Assert the corresponding reset 0 = DISABLE 1 = ENABLE
13	0x0	RST_OVRD_FCPU_RAIL: 1 = Assert the corresponding reset 0 = DISABLE 1 = ENABLE
12	0x0	RST_OVRD_FCPU_NC: 1 = Assert the corresponding reset 0 = DISABLE 1 = ENABLE
11	0x0	RST_OVRD_FCPU_3: 1 = Assert the corresponding reset 0 = DISABLE 1 = ENABLE
10	0x0	RST_OVRD_FCPU_2: 1 = Assert the corresponding reset 0 = DISABLE 1 = ENABLE
9	0x0	RST_OVRD_FCPU_1: 1 = Assert the corresponding reset 0 = DISABLE 1 = ENABLE
8	0x0	RST_OVRD_FCPU_0: 1 = Assert the corresponding reset 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
7	0x0	EN_RST_CG_OVRD_SCPU_NC: 1 = Select the OVRD controls in this register 0 = DISABLE 1 = ENABLE
6	0x0	EN_RST_CG_OVRD_SCPU_0: 1 = Select the OVRD controls in this register 0 = DISABLE 1 = ENABLE
5	0x0	EN_RST_CG_OVRD_FCPU_RAIL: 1 = Select the OVRD controls in this register 0 = DISABLE 1 = ENABLE
4	0x0	EN_RST_CG_OVRD_FCPU_NC: 1 = Select the OVRD controls in this register 0 = DISABLE 1 = ENABLE
3	0x0	EN_RST_CG_OVRD_FCPU_3: 1 = Select the OVRD controls in this register 0 = DISABLE 1 = ENABLE
2	0x0	EN_RST_CG_OVRD_FCPU_2: 1 = Select the OVRD controls in this register 0 = DISABLE 1 = ENABLE
1	0x0	EN_RST_CG_OVRD_FCPU_1: 1 = Select the OVRD controls in this register 0 = DISABLE 1 = ENABLE
0	0x0	EN_RST_CG_OVRD_FCPU_0: 1 = Select the OVRD controls in this register 0 = DISABLE 1 = ENABLE

5.7.106 CLK_RST_CONTROLLER_RST_CPU_CMPLX_SET_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

Offset: 0x340 │ Read/Write: R/W │ Reset: 0x2000feef (0bx010xxx00000000011111110111011111)

Bit	Reset	Description
30	0x0	SET_PRESETDBG: 1 = Assert nPRESETDBG to the debug APB interface. 0 = DISABLE 1 = ENABLE
29	0x1	SET_SCURESET: 1 = Assert reset to the whole nonCPU region of the CPU. 0 = DISABLE 1 = ENABLE
28	0x0	SET_PERIPHRESET: Reserved.
24	0x0	SET_L2RESET: 1 = Assert nL2RESET to the CPU. 0 = DISABLE 1 = ENABLE
23	0x0	SET_CXRESET3: 1 = Assert nCXRESET to CPU3. 0 = DISABLE 1 = ENABLE
22	0x0	SET_CXRESET2: 1 = Assert nCXRESET to CPU2. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
21	0x0	SET_CXRESET1: 1 = Assert nCXRESET to CPU1. 0 = DISABLE 1 = ENABLE
20	0x0	SET_CXRESET0: 1 = Assert nCXRESET to CPU0. 0 = DISABLE 1 = ENABLE
19	0x0	SET_CORERESET3: 1 = Assert nCORERESET to CPU3. 0 = DISABLE 1 = ENABLE
18	0x0	SET_CORERESET2: 1 = Assert nCORERESET to CPU2. 0 = DISABLE 1 = ENABLE
17	0x0	SET_CORERESET1: 1 = Assert nCORERESET to CPU1. 0 = DISABLE 1 = ENABLE
16	0x0	SET_CORERESET0: 1 = Assert nCORERESET to CPU0. 0 = DISABLE 1 = ENABLE
15	0x1	SET_DBGRESET3: 1 = Assert nDBGRESET to CPU3. 0 = DISABLE 1 = ENABLE
14	0x1	SET_DBGRESET2: 1 = Assert nDBGRESET to CPU2. 0 = DISABLE 1 = ENABLE
13	0x1	SET_DBGRESET1: 1 = Assert nDBGRESET to CPU1. 0 = DISABLE 1 = ENABLE
12	0x1	SET_DBGRESET0: 1 = Assert nDBGRESET to CPU0. 0 = DISABLE 1 = ENABLE
11	0x1	SET_WDRESET3: Reserved.
10	0x1	SET_WDRESET2: Reserved.
9	0x1	SET_WDRESET1: Reserved.
8	0x0	SET_WDRESET0: Reserved.
7	0x1	SET_DERESET3: Reserved.
6	0x1	SET_DERESET2: Reserved.
5	0x1	SET_DERESET1: Reserved.
4	0x0	SET_DERESET0: Reserved.
3	0x1	SET_CPURESET3: 1 = assert nCPUPORRESET to CPU3. 0 = DISABLE 1 = ENABLE
2	0x1	SET_CPURESET2: 1 = assert nCPUPORRESET to CPU2. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x1	SET_CPURESET1: 1 = assert nCPUPORRESET to CPU1. 0 = DISABLE 1 = ENABLE
0	0x1	SET_CPURESET0: 1 = assert nCPUPORRESET to CPU0. 0 = DISABLE 1 = ENABLE

5.7.107 CLK_RST_CONTROLLER_RST_CPU_CMPLX_CLR_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

Offset: 0x344 │ Read/Write: R/W │ Reset: 0x2000feef (0bx010xxx00000000011111111011101111)

Bit	Reset	Description
30	0x0	CLR_PRESETDBG: 1 = De-assert nPRESETDBG to the CoreSight. 0 = DISABLE 1 = ENABLE
29	0x1	CLR_NONCPURESET: 1 = De-assert reset to the whole nonCPU region of the CPU. 0 = DISABLE 1 = ENABLE
28	0x0	CLR_PERIPHRESET: Reserved.
24	0x0	CLR_L2RESET: 1 = De-assert nL2RESET to CPU. 0 = DISABLE 1 = ENABLE
23	0x0	CLR_CXRESET3: 1 = De-assert nCXRESET to CPU30 = DISABLE 1 = ENABLE
22	0x0	CLR_CXRESET2: 1 = De-assert nCXRESET to CPU2. 0 = DISABLE 1 = ENABLE
21	0x0	CLR_CXRESET1: 1 = De-assert nCXRESET to CPU1. 0 = DISABLE 1 = ENABLE
20	0x0	CLR_CXRESET0: 1 = De-assert nCXRESET to CPU0. 0 = DISABLE 1 = ENABLE
19	0x0	CLR_CORERESET3: 1 = De-assert nCORERESET to CPU3. 0 = DISABLE 1 = ENABLE
18	0x0	CLR_CORERESET2: 1 = De-assert nCORERESET to CPU2. 0 = DISABLE 1 = ENABLE
17	0x0	CLR_CORERESET1: 1 = De-assert nCORERESET to CPU1. 0 = DISABLE 1 = ENABLE
16	0x0	CLR_CORERESET0: 1 = De-assert nCORERESET to CPU0. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
15	0x1	CLR_DBGRESET3: 1 = De-assert nDBGRESET to CPU3. 0 = DISABLE 1 = ENABLE
14	0x1	CLR_DBGRESET2: 1 = De-assert nDBGRESET to CPU2. 0 = DISABLE 1 = ENABLE
13	0x1	CLR_DBGRESET1: 1 = De-assert nDBGRESET to CPU1. 0 = DISABLE 1 = ENABLE
12	0x1	CLR_DBGRESET0: 1 = De-assert nDBGRESET to CPU0. 0 = DISABLE 1 = ENABLE
11	0x1	CLR_WDRESET3: Reserved.
10	0x1	CLR_WDRESET2: Reserved.
9	0x1	CLR_WDRESET1: Reserved.
8	0x0	CLR_WDRESET0: Reserved.
7	0x1	CLR_DERESET3: Reserved.
6	0x1	CLR_DERESET2: Reserved.
5	0x1	CLR_DERESET1: Reserved.
4	0x0	CLR_DERESET0: Reserved.
3	0x1	CLR_CPURESET3: 1 = De-assert nCPUPORRESET to CPU3. 0 = DISABLE 1 = ENABLE
2	0x1	CLR_CPURESET2: 1 = De-assert nCPUPORRESET to CPU2. 0 = DISABLE 1 = ENABLE
1	0x1	CLR_CPURESET1: 1 = De-assert nCPUPORRESET to CPU1. 0 = DISABLE 1 = ENABLE
0	0x1	CLR_CPURESET0: 1 = De-assert nCPUPORRESET to CPU0. 0 = DISABLE 1 = ENABLE

5.7.108 CLK_RST_CONTROLLER_CLK_CPU_CMPLX_SET_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

Offset: 0x348 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxxxxxx)

Bit	Reset	Description
11	0x0	SET_CPU3_CLK_STP: 1 = Assert CPU3 clock stop 0 = DISABLE 1 = ENABLE
10	0x0	SET_CPU2_CLK_STP: 1 = Assert CPU2 clock stop 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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

5.7.109 CLK_RST_CONTROLLER_CLK_CPU_CMPLX_CLR_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

Offset: 0x34c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxxxxxx)

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

5.7.110 CLK_RST_CONTROLLER_RST_DEVICES_V_0

Expanded Register Set

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

Offset: 0x358 │ Read/Write: R/W │ Reset: 0xff81ffeX (0b111111111xxxxxx111111111111x1xxx)

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

Bit	R/W	Reset	Description
			0 = DISABLE 1 = ENABLE
12	RW	ENABLE	SWR_DAM0_RST: Reset DAM0 0 = DISABLE 1 = ENABLE
11	RW	ENABLE	SWR_APBIF_RST: Reset APBIF 0 = DISABLE 1 = ENABLE
10	RW	ENABLE	SWR_AUDIO_RST: Reset AUDIO 0 = DISABLE 1 = ENABLE
9	RW	ENABLE	SWR_SPI6_RST: Reset SPI6 0 = DISABLE 1 = ENABLE
8	RW	ENABLE	SWR_SPI5_RST: Reset SPI5 0 = DISABLE 1 = ENABLE
7	RW	ENABLE	SWR_I2C4_RST: Reset I2C4 0 = DISABLE 1 = ENABLE
6	RW	ENABLE	SWR_I2S4_RST: Reset I2S4 0 = DISABLE 1 = ENABLE
5	RW	ENABLE	SWR_I2S3_RST: Reset I2S3 0 = DISABLE 1 = ENABLE
3	RW	ENABLE	SWR_MSELECT_RST: Reset MSELECT 0 = DISABLE 1 = ENABLE
1	RO	X	SWR_CPULP_RST: Reset CPUG.
0	RO	X	SWR_CPUG_RST: Reserved.

5.7.111 CLK_RST_CONTROLLER_RST_DEVICES_W_0

Offset: 0x35c │ Read/Write: R/W │ Reset: 0x1f407fff (0bxxx11111x10xxxxxx111111111111111)

Bit	Reset	Description
28	ENABLE	SWR_XUSB_SS_RST: Reset XUSB_SS logic. 0 = DISABLE 1 = ENABLE
27	ENABLE	SWR_DVFS_RST: Reset CLDVFS 0 = DISABLE 1 = ENABLE
26	ENABLE	SWR_ADX0_RST: Reset ADX0 0 = DISABLE 1 = ENABLE
25	ENABLE	SWR_AMX0_RST: Reset AMX0 0 = DISABLE 1 = ENABLE
21	DISABLE	SWR_ENTROPY_RST: Reset Entropy logic. 0 = DISABLE 1 = ENABLE
14	ENABLE	SWR_XUSB_PADCTL_RST: Reset XUSB PADCTL logic. IOBIST control has clock enable only.

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
13	0x1	SWR_RESERVED10_RST: Reserved
12	0x1	SWR_RESERVED9_RST: Reserved
11	0x1	SWR_RESERVED8_RST: Reserved
10	0x1	SWR_RESERVED7_RST: Reserved
9	0x1	SWR_RESERVED6_RST: Reserved
8	ENABLE	SWR_CEC_RST: Reset CEC 0 = DISABLE 1 = ENABLE
7	0x1	SWR_RESERVED5_RST: Reserved
6	0x1	SWR_RESERVED4_RST: Reserved
5	0x1	SWR_RESERVED3_RST: Reserved
4	0x1	SWR_RESERVED2_RST: Reserved
3	0x1	SWR_RESERVED1_RST: Reserved
2	0x1	SWR_RESERVED0_RST: Reserved
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

5.7.112 CLK_RST_CONTROLLER_CLK_OUT_ENB_V_0

Offset: 0x360 │ Read/Write: R/W │ Reset: 0x00400000 (0b0000000001xxxxx00000000000000x00)

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

Bit	Reset	Description
22	ENABLE	CLK_ENB_SPDIF_DOUBLER: Enable S/PDIF audio sync clock 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_SPI6: Enable clock to SPI6 0 = DISABLE 1 = ENABLE
8	DISABLE	CLK_ENB_SPI5: Enable clock to SPI5 0 = DISABLE 1 = ENABLE
7	DISABLE	CLK_ENB_I2C4: Enable clock to I2C4 0 = DISABLE 1 = ENABLE
6	DISABLE	CLK_ENB_I2S4: Enable clock to I2S4 0 = DISABLE 1 = ENABLE
5	DISABLE	CLK_ENB_I2S3: Enable clock to I2S3 0 = DISABLE 1 = ENABLE
4	DISABLE	CLK_ENB_TSENSOR: Enable clock to TSENSOR 0 = DISABLE 1 = ENABLE
3	DISABLE	CLK_ENB_MSELECT: Enable clock to MSELECT 0 = DISABLE 1 = ENABLE
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

5.7.113 CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0

Offset: 0x364 | Read/Write: R/W | Reset: 0x002000fc (0bxx000000x01000000x00000011111100)

Bit	Reset	Description
29	DISABLE	CLK_ENB_EMC_LATENCY: Enable clock to EMC latency 0 = DISABLE 1 = ENABLE
28	DISABLE	CLK_ENB_XUSB_SS: Enable clock to XUSB_SS 0 = DISABLE 1 = ENABLE
27	DISABLE	CLK_ENB_DVFS: Enable clock to CLDVFS 0 = DISABLE 1 = ENABLE
26	DISABLE	CLK_ENB_ADX0: Enable clock to ADX0 0 = DISABLE 1 = ENABLE
25	DISABLE	CLK_ENB_AMX0: Enable clock to AMX0 0 = DISABLE 1 = ENABLE
21	ENABLE	CLK_ENB_ENTROPY: Enable clock to ENTROPY. 0 = DISABLE 1 = ENABLE
20	DISABLE	CLK_ENB_DSIB_LP: Enable clock to DSIB LP. 0 = DISABLE 1 = ENABLE
19	DISABLE	CLK_ENB_DSIA_LP: Enable clock to DSIA LP. 0 = DISABLE 1 = ENABLE
18	DISABLE	CLK_ENB_CILE: Enable clock to CSI CILE. 0 = DISABLE 1 = ENABLE
17	DISABLE	CLK_ENB_CILCD: Enable clock to CSI CILC and CILD. 0 = DISABLE 1 = ENABLE
16	DISABLE	CLK_ENB_CILAB: Enable clock to CSI CILA and CILB. 0 = DISABLE 1 = ENABLE
15	DISABLE	CLK_ENB_XUSB: Enable clock to XUSB. IOBIST control has clock enable only. 0 = DISABLE 1 = ENABLE
13	DISABLE	CLK_ENB_MIPI_IOBIST: Enable clock to MIPI IOBIST. 0 = DISABLE 1 = ENABLE
12	DISABLE	CLK_ENB_SATA_IOBIST: Enable clock to SATA IOBIST. 0 = DISABLE 1 = ENABLE
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

Bit	Reset	Description
8	DISABLE	CLK_ENB_CEC: Enable clock to CEC. 0 = DISABLE 1 = ENABLE
7	0x1	CLK_ENB_PCIERX5: Enable clock to PCIERX5. CLK_ENB_PCIE is also the master control for all PCIERX* clocks.
6	0x1	CLK_ENB_PCIERX4: Enable clock to PCIERX4. CLK_ENB_PCIE is also the master control for all PCIERX* clocks.
5	0x1	CLK_ENB_PCIERX3: Enable clock to PCIERX3. CLK_ENB_PCIE is also the master control for all PCIERX* clocks.
4	0x1	CLK_ENB_PCIERX2: Enable clock to PCIERX2. CLK_ENB_PCIE is also the master control for all PCIERX* clocks.
3	0x1	CLK_ENB_PCIERX1: Enable clock to PCIERX1. CLK_ENB_PCIE is also the master control for all PCIERX* clocks.
2	0x1	CLK_ENB_PCIERX0: Enable clock to PCIERX0. CLK_ENB_PCIE is also the master control for all PCIERX* clocks.
1	DISABLE	CLK_ENB_RESERVED0: Unused clock enable for satacoldrstn 0 = DISABLE 1 = ENABLE
0	DISABLE	CLK_ENB_HDA2HDMICODEC: Enable clock to HDA2HDMICODEC (no dedicated module or clock). 0 = DISABLE 1 = ENABLE

5.7.114 CLK_RST_CONTROLLER_CCLKG_BURST_POLICY_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

Bit	R/W	Reset	Description
31:28	RW	0x1	CPU_STATE: 0000=32 kHz Clock source; 0001=IDLE Clock Source; 001X=Run clock source; 01XX=IRQ Clock Source; 1XXX=FIQ Clock Source 0 = STDBY 1 = IDLE 2 = RUN 4 = IRQ 8 = FIQ
27	RW	0x0	COP_AUTO_CWAKEUP_FROM_FIQ: 0 = NOP ; 1 = Burst on COP FIQ
26	RW	0x0	CPU_AUTO_CWAKEUP_FROM_FIQ: 0 = NOP ; 1 = Burst on CPU FIQ
25	RW	0x0	COP_AUTO_CWAKEUP_FROM_IRQ: 0 = NOP ; 1 = Burst on COP IRQ
24	RW	0x0	CPU_AUTO_CWAKEUP_FROM_IRQ: 0 = NOP ; 1 = Burst on CPU IRQ
23	RO	X	TSENSOR_SLOWDOWN: 0 = Normal ; 1 = cpug_clk/2 triggered by the temperature sensor.
16	RO	X	CCLK_RESERVED: Reserved. cpulp uses this bit for div2 bypass.
15:12	RW	0x0	CWAKEUP_FIQ_SOURCE: 0000 = clk_m, 0001 = pllC_out0, 0010 = clk_s, 0011 = pllM_out0, 0100 = pllP_out0, 0101 = pllP_out4, 0110 = pllC2_out0, 0111 = pllC3_out0, 1000 = PLLX_out0 (low jitter), 1001 = dvfs_cpu_clk, 1110 = PLLX_out0, 1111 = dvfs_cpu_clk, (low jitter) 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4

Bit	R/W	Reset	Description
			6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = DVFS_CPU_CLK 14 = PLLX_OUT0 15 = DVFS_CPU_CLK_LJ
11:8	RW	0x0	CWAKEUP_IRQ_SOURCE: Same definitions as CWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = DVFS_CPU_CLK 14 = PLLX_OUT0 15 = DVFS_CPU_CLK_LJ
7:4	RW	0x0	CWAKEUP_RUN_SOURCE: Same definitions as CWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = DVFS_CPU_CLK 14 = PLLX_OUT0 15 = DVFS_CPU_CLK_LJ
3:0	RW	0x0	CWAKEUP_IDLE_SOURCE: Same definitions as CWAKEUP_FIQ 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = DVFS_CPU_CLK 14 = PLLX_OUT0 15 = DVFS_CPU_CLK_LJ

5.7.115 CLK_RST_CONTROLLER_SUPER_CCLKG_DIVIDER_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

Bit	Reset	Description
31	DISABLE	SUPER_CDIV_ENB: 0 = Disable divider. 0 = DISABLE 1 = ENABLE
30	DISABLE	SUPER_CDIV_USE_THERM_CONTROLS: 1 = Use thermal controls for pulse skipper 0 = DISABLE 1 = ENABLE
28	DISABLE	CCLK_INVERT_DCD: 1 = Enable inversion of DCD (duty-cycle distortion), 0 = Disable inversion of DCD 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
		See “Invert Duty-Cycle Distortion Control” in this section for more information.
27	0x0	SUPER_CDIV_DIS_FROM_COP_FIQ: 0 = enable FIQ, disable FIQ.
26	0x0	SUPER_CDIV_DIS_FROM_CPU_FIQ: 0 = enable FIQ, disable FIQ.
25	0x0	SUPER_CDIV_DIS_FROM_COP_IRQ: 0 = enable IRQ, disable IRQ.
24	0x0	SUPER_CDIV_DIS_FROM_CPU_IRQ: 0 = enable IRQ, disable IRQ.
23:16	0x0	CCLK_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_CDIV_DIVISOR: Actual value = n + 1.

5.7.116 CLK_RST_CONTROLLER_CCLKLP_BURST_POLICY_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

This is not a glitch-less switch. Do not change the register field while CPULP is using PLLX as a clock source.

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

Bit	R/W	Reset	Description
31:28	RW	0x1	CPULP_STATE: 0000=32 kHz Clock source; 0001=IDLE Clock Source; 001X=Run clock source; 01XX=IRQ Clock Source; 1XXX=FIQ Clock Source 0 = STDBY 1 = IDLE 2 = RUN 4 = IRQ 8 = FIQ
27	RW	0x0	COP_AUTO_CWAKEUP_FROM_FIQ: 0 = NOP ; 1=Burst on COP FIQ
26	RW	0x0	CPU_AUTO_CWAKEUP_FROM_FIQ: 0 = NOP ; 1=Burst on CPU FIQ
25	RW	0x0	COP_AUTO_CWAKEUP_FROM_IRQ: 0 = NOP ; 1=Burst on COP IRQ
24	RW	0x0	CPU_AUTO_CWAKEUP_FROM_IRQ: 0 = NOP ; 1=Burst on CPU IRQ
23	RO	X	RESERVED: Reserved for tsensor_slowdown status for CPUG.
16	RW	0x0	PLLX_DIV2_BYPASS_LP: 0 = PLLX/2 ; 1 = PLLX/1 for source=1000
15:12	RW	0x0	CWAKEUP_FIQ_SOURCE: 0000 = clk_m, 0001 = pllC_out0, 0010 = clk_s, 0011 = pllM_out0, 0100 = pllP_out0, 0101 = pllP_out4, 0110 = pllC2_out0, 0111 = pllC3_out0, 1000 = PLLX_out0, 1001 = Reserved, 1110 = PLLX_out0, 1111 = Reserved 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0_LJ 8 = PLLX_OUT0 9 = RESERVED9 14 = PLLX_OUT0 15 = RESERVED15
11:8	RW	0x0	CWAKEUP_IRQ_SOURCE: Same definitions as CWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT0 2 = CLKS

Bit	R/W	Reset	Description
			3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = RESERVED9 14 = PLLX_OUT0 15 = RESERVED15
7:4	RW	0x0	CWAKEUP_RUN_SOURCE: Same definitions as CWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = RESERVED9 14 = PLLX_OUT0 15 = RESERVED15
3:0	RW	0x0	CWAKEUP_IDLE_SOURCE: Same definitions as CWAKEUP_FIQ 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = RESERVED9 14 = PLLX_OUT0 15 = RESERVED15

5.7.117 CLK_RST_CONTROLLER_SUPER_CCLKLP_DIVIDER_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

Bit	Reset	Description
31	DISABLE	SUPER_CDIV_ENB: 0 = Disable divider. 0 = DISABLE 1 = ENABLE
30	DISABLE	RESERVED: Reserved for cpug override of thermal control 0 = DISABLE 1 = ENABLE
28	DISABLE	CCLK_INVERT_DCD: 1 = Enable inversion of DCD (duty-cycle distortion), 0 = Disable inversion of DCD 0 = DISABLE 1 = ENABLE See “Invert Duty-Cycle Distortion Control” in this section for more information.
27	0x0	SUPER_CDIV_DIS_FROM_COP_FIQ: 0 = enable FIQ, disable FIQ.
26	0x0	SUPER_CDIV_DIS_FROM_CPU_FIQ: 0 = enable FIQ, disable FIQ.
25	0x0	SUPER_CDIV_DIS_FROM_COP_IRQ: 0 = enable IRQ, disable IRQ.
24	0x0	SUPER_CDIV_DIS_FROM_CPU_IRQ: 0 = enable IRQ, disable IRQ.
23:16	0x0	CCLK_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

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

5.7.118 CLK_RST_CONTROLLER_CLK_CPUG_CMPLX_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details..

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

Note: Because the CPUG clock can be selected from 1-of-9 clock sources (refer to the CCLKG_BURST_POLICY register), it is the user’s responsibility to not select a bridge divide-down CPUG clock ratio that will exceed 1/4 of the maximum CPUG frequency supported. The maximum CPUG frequency is 1.1 GHz.

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

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

5.7.119 CLK_RST_CONTROLLER_CLK_CPULP_CMPLX_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

Note: Because the CPUG clock can be selected from 1-of-9 clock sources (refer to the CCLKLP_BURST_POLICY register), it is the user’s responsibility to not select a bridge divide-down CPULG clock ratio that will exceed 1/4 of the maximum CPULG frequency supported. The maximum CPULG frequency is 1.1 GHz.

Offset: 0x37c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx)

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

5.7.120 CLK_RST_CONTROLLER_CPU_SOFTRST_CTRL_0

Earlier Tegra chips had a single state-machine that took care of sequencing reset (delaying reset de-assertion based on a timer value) in case of a flow-controller request or a WatchDog Timer (WDT) expiry request. This changed in Tegra K1 devices because a flow-controller initiated request has more to do. The timer control of reset de-assertion for WDT initiated requests is in the same place while the new controls have been added to the new registers.

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

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

5.7.121 CLK_RST_CONTROLLER_CPU_SOFTRST_CTRL1_0

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

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

5.7.122 CLK_RST_CONTROLLER_CPU_SOFTRST_CTRL2_0

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

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

5.7.123 CLK_RST_CONTROLLER_CLK_SOURCE_MSELECT_0

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

Bit	Reset	Description
31:29	CLK_M	MSELECT_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 101 = clk_s, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 5 = CLK_S 6 = CLK_M
7:0	0x0	MSELECT_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.124 CLK_RST_CONTROLLER_CLK_SOURCE_TSENSOR_0

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

Bit	Reset	Description
31:29	CLK_S	TSENSOR_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = clk_m, 110 = clk_s 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0

Bit	Reset	Description
		4 = CLK_M 6 = CLK_S
7:0	0x0	TSENSOR_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.125 CLK_RST_CONTROLLER_CLK_SOURCE_I2S3_0

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

Bit	Reset	Description
31:29	CLK_M	I2S3_CLK_SRC: 000 = pllA_out0, 010 = audio SYNC_CLK 1x or 2x, 100 = pllP_out0, 110 = clk_m 0 = PLLA_OUT0 2 = SYNC_CLK 4 = PLLP_OUT0 6 = CLK_M
28	0x1	I2S3_MASTER_CLKEN: Reserved.
7:0	0x0	I2S3_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.126 CLK_RST_CONTROLLER_CLK_SOURCE_I2S4_0

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

Bit	Reset	Description
31:29	CLK_M	I2S4_CLK_SRC: 000 = pllA_out0, 010 = audio SYNC_CLK 1x or 2x, 100 = pllP_out0, 110 = clk_m 0 = PLLA_OUT0 2 = SYNC_CLK 4 = PLLP_OUT0 6 = CLK_M
28	0x1	I2S4_MASTER_CLKEN: Reserved.
7:0	0x0	I2S4_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.127 CLK_RST_CONTROLLER_CLK_SOURCE_I2C4_0

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

Bit	Reset	Description
31:29	CLK_M	I2C4_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
15:0	0x0	I2C4_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 1.0x)

5.7.128 CLK_RST_CONTROLLER_CLK_SOURCE_SPI5_0

Offset: 0x3c8 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SPI5_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0

Bit	Reset	Description
		1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	SPI5_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.129 CLK_RST_CONTROLLER_CLK_SOURCE_SPI6_0

Offset: 0x3cc │ Read/Write: R/W │ Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SPI6_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	SPI6_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.130 CLK_RST_CONTROLLER_CLK_SOURCE_AUDIO_0

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

Bit	Reset	Description
31:29	CLK_M	AUDIO_CLK_SRC: 000 = pllA_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = clk_m, 111 = audio_clk_src_alt 0 = PLLA_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = CLK_M 7 = CLK_SRC_ALT
28	0x1	AUDIO_MASTER_CLKEN: Reserved.
20	ENABLE	AUDIO_CLK_SRC_DIS: 0 = Enable audio_clk_src_alt 0 = ENABLE 1 = DISABLE
19:16	SPDIFIN	AUDIO_CLK_SRC_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S0 bit clock. 0010 = I2S1 bit clock. 0011 = I2S2 bit clock. 0100 = I2S3 bit clock. 0101 = I2S4 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S0 2 = I2S1 3 = I2S2 4 = I2S3 5 = I2S4 6 = PLLA_OUT0 7 = EXT_VIMCLK
7:0	0x0	AUDIO_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.131 CLK_RST_CONTROLLER_CLK_SOURCE_DAM0_0

Offset: 0x3d8 │ Read/Write: R/W │ Reset: 0xd0000000 (0b1101xxxxxxx00000xxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	DAM0_CLK_SRC: 000 = pllA_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = clk_m, 111 = dam0_clk_src_alt 0 = PLLA_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = CLK_M 7 = CLK_SRC_ALT
28	0x1	DAM0_MASTER_CLKEN: Reserved.
20	ENABLE	DAM0_CLK_SRC_DIS: 0 = Enable dam0_clk_src_alt 0 = ENABLE 1 = DISABLE
19:16	SPDIFIN	DAM0_CLK_SRC_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S0 bit clock. 0010 = I2S1 bit clock. 0011 = I2S2 bit clock. 0100 = I2S3 bit clock. 0101 = I2S4 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S0 2 = I2S1 3 = I2S2 4 = I2S3 5 = I2S4 6 = PLLA_OUT0 7 = EXT_VIMCLK
7:0	0x0	DAM0_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.132 CLK_RST_CONTROLLER_CLK_SOURCE_DAM1_0

Offset: 0x3dc │ Read/Write: R/W │ Reset: 0xd0000000 (0b1101xxxxxxx00000xxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	DAM1_CLK_SRC: 000 = pllA_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = clk_m, 111 = dam1_clk_src_alt 0 = PLLA_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = CLK_M 7 = CLK_SRC_ALT
28	0x1	DAM1_MASTER_CLKEN: Reserved.
20	ENABLE	DAM1_CLK_SRC_DIS: 0 = Enable dam1_clk_src_alt 0 = ENABLE 1 = DISABLE
19:16	SPDIFIN	DAM1_CLK_SRC_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S0 bit clock. 0010 = I2S1 bit clock. 0011 = I2S2 bit clock. 0100 = I2S3 bit clock. 0101 = I2S4 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S0 2 = I2S1 3 = I2S2 4 = I2S3 5 = I2S4 6 = PLLA_OUT0 7 = EXT_VIMCLK
7:0	0x0	DAM1_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.133 CLK_RST_CONTROLLER_CLK_SOURCE_DAM2_0

Offset: 0x3e0 | Read/Write: R/W | Reset: 0xd0000000 (0b1101xxxxxxx00000xxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	DAM2_CLK_SRC: 000 = pllA_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = clk_m, 111 = dam2_clk_src_alt 0 = PLLA_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = CLK_M 7 = CLK_SRC_ALT
28	0x1	DAM2_MASTER_CLKEN: Reserved.
20	ENABLE	DAM2_CLK_SRC_DIS: 0 = Enable dam2_clk_src_alt 0 = ENABLE 1 = DISABLE
19:16	SPDIFIN	DAM2_CLK_SRC_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S0 bit clock. 0010 = I2S1 bit clock. 0011 = I2S2 bit clock. 0100 = I2S3 bit clock. 0101 = I2S4 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S0 2 = I2S1 3 = I2S2 4 = I2S3 5 = I2S4 6 = PLLA_OUT0 7 = EXT_VIMCLK
7:0	0x0	DAM2_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.134 CLK_RST_CONTROLLER_CLK_SOURCE_HDA2CODEC_2X_0

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

Bit	Reset	Description
31:29	CLK_M	HDA2CODEC_2X_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	HDA2CODEC_2X_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.135 CLK_RST_CONTROLLER_CLK_SOURCE_ACTMON_0

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

Bit	Reset	Description
31:29	CLK_M	ACTMON_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = clk_s, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = CLK_S 6 = CLK_M
7:0	0x0	ACTMON_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.136 CLK_RST_CONTROLLER_CLK_SOURCE_EXTPERIPH1_0

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

Bit	Reset	Description
31:29	CLK_M	EXTPERIPH1_CLK_SRC: 000 = pllA_out0, 001 = clk_s, 010 = pllP_out0, 011 = clk_m, 100 = pllE_out0 0 = PLLA_OUT0 1 = CLK_S 2 = PLLP_OUT0 3 = CLK_M 4 = PLLE_OUT0
7:0	0x0	EXTPERIPH1_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.137 CLK_RST_CONTROLLER_CLK_SOURCE_EXTPERIPH2_0

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

Bit	Reset	Description
31:29	CLK_M	EXTPERIPH2_CLK_SRC: 000 = pllA_out0, 001 = clk_s, 010 = pllP_out0, 011 = clk_m, 100 = pllE_out0 0 = PLLA_OUT0 1 = CLK_S 2 = PLLP_OUT0 3 = CLK_M 4 = PLLE_OUT0
7:0	0x0	EXTPERIPH2_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.138 CLK_RST_CONTROLLER_CLK_SOURCE_EXTPERIPH3_0

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

Bit	Reset	Description
31:29	CLK_M	EXTPERIPH3_CLK_SRC: 000 = pllA_out0, 001 = clk_s, 010 = pllP_out0, 011 = clk_m, 100 = pllE_out0 0 = PLLA_OUT0 1 = CLK_S 2 = PLLP_OUT0 3 = CLK_M 4 = PLLE_OUT0
7:0	0x0	EXTPERIPH3_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.139 CLK_RST_CONTROLLER_CLK_SOURCE_I2C_SLOW_0

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

Bit	Reset	Description
31:29	CLK_M	I2C_SLOW_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = clk_s, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = CLK_S 6 = CLK_M
7:0	0x0	I2C_SLOW_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.140 CLK_RST_CONTROLLER_CLK_SOURCE_SYS_0

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

Offset: 0x400 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

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

5.7.141 CLK_RST_CONTROLLER_CLK_SOURCE_SOR0_0

Offset: 0x414 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxx00xxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SOR0_CLK_SRC: 000 = pllP_out0, 001 = pllM_out0, 010 = pllD_out0, 011 = pllA_out0, 100 = pllC_out0, 101 = PLLD2_out0, 110 = clk_m, 111 = 1'b0 Non sequential mapping used to preserve 2 MSBS to match legacy source select. 0 = PLLP_OUT0 1 = PLLM_OUT0 2 = PLLD_OUT0 3 = PLLA_OUT0 4 = PLLC_OUT 5 = PLLD2_OUT2 6 = CLK_M
15	0x0	SOR0_CLK_SEL1: 0 = safe clock 24 MHz, 1 = LVDS pixel clock
14	0x0	SOR0_CLK_SEL0: 0 = mux output of safe_clk or LVDS pixel clock, 1 = eDP Macro output
7:0	0x0	I2C_SLOW_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.142 CLK_RST_CONTROLLER_CLK_SOURCE_SATA_OOB_0

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

Bit	Reset	Description
31:29	CLK_M	SATA_OOB_CLK_SRC: 000 = pllP_out0, 010 = pllC_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 2 = PLLC_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	SATA_OOB_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.143 CLK_RST_CONTROLLER_CLK_SOURCE_SATA_0

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

Bit	Reset	Description
31:29	CLK_M	SATA_CLK_SRC: 000 = pllP_out0, 010 = pllC_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 2 = PLLC_OUT0 4 = PLLM_OUT0 6 = CLK_M
24	ENABLE	SATA_AUX_CLK_ENB: ENABLE SATA Tx/Rx clocks 0 = DISABLE 1 = ENABLE
7:0	0x0	SATA_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.144 CLK_RST_CONTROLLER_CLK_SOURCE_HDA_0

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

Bit	Reset	Description
31:29	CLK_M	HDA_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	HDA_CLK_DIVISOR: N = Divide by (n+1) (lsb denotes 0.5x)

5.7.145 CLK_RST_CONTROLLER_RST_DEV_V_SET_0

Offset: 0x430 | Read/Write: R/W | Reset: 0xff81ffeX (0b111111111xxxxxx111111111111x1xxx)

Bit	R/W	Reset	Description
29	RW	0x1	SET_HDA_RST: Set reset for the HDA 0 = DISABLE 1 = ENABLE
28	RW	0x1	SET_SATA_RST: Set reset for the SATA 0 = DISABLE 1 = ENABLE
23	RW	0x1	SET_ACTMON_RST: Set reset for the ACTMON 0 = DISABLE 1 = ENABLE
16	RW	0x1	SET_ATOMICS_RST: Set reset for the ATOMICS 0 = DISABLE 1 = ENABLE
15	RW	0x1	SET_HDA2CODEC_2X_RST: Set reset for the HDA2CODEC_2X 0 = DISABLE 1 = ENABLE
14	RW	0x1	SET_DAM2_RST: Set reset for the DAM2 0 = DISABLE 1 = ENABLE
13	RW	0x1	SET_DAM1_RST: Set reset for the DAM1 0 = DISABLE 1 = ENABLE
12	RW	0x1	SET_DAM0_RST: Set reset for the DAM0 0 = DISABLE 1 = ENABLE
11	RW	0x1	SET_APBIF_RST: Set reset for the APBIF 0 = DISABLE 1 = ENABLE
10	RW	0x1	SET_AUDIO_RST: Set reset for the AUDIO 0 = DISABLE 1 = ENABLE
9	RW	0x1	SET_SPI6_RST: Set reset for the SPI6 0 = DISABLE 1 = ENABLE
8	RW	0x1	SET_SPI5_RST: Set reset for the SPI5 0 = DISABLE 1 = ENABLE
7	RW	0x1	SET_I2C4_RST: Set reset for the I2C4

Bit	R/W	Reset	Description
			0 = DISABLE 1 = ENABLE
6	RW	0x1	SET_I2S4_RST: Set reset for the I2S4 0 = DISABLE 1 = ENABLE
5	RW	0x1	SET_I2S3_RST: Set reset for the I2S3 0 = DISABLE 1 = ENABLE
3	RW	0x1	SET_MSELECT_RST: Set reset for the MSELECT 0 = DISABLE 1 = ENABLE
1	RO	X	SET_CPULP_RST: Set reset for the CPULP. [CCLK Multi-Address]. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE
0	RO	X	SET_CPUG_RST: Set reset for the CPUG. [CCLK Multi-Address]. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE

5.7.146 CLK_RST_CONTROLLER_RST_DEV_V_CLR_0

Offset: 0x434 | Read/Write: R/W | Reset: 0xff81ffeX (0b111111111xxxxxx111111111111x1xxx)

Bit	R/W	Reset	Description
29	RW	0x1	CLR_HDA_RST: Clear reset for HDA 0 = DISABLE 1 = ENABLE
28	RW	0x1	CLR_SATA_RST: Clear reset for SATA 0 = DISABLE 1 = ENABLE
23	RW	0x1	CLR_ACTMON_RST: Clear reset for ACTMON 0 = DISABLE 1 = ENABLE
16	RW	0x1	CLR_ATOMICS_RST: Clear reset for ATOMICS 0 = DISABLE 1 = ENABLE
15	RW	0x1	CLR_HDA2CODEC_2X_RST: Clear reset for HDA2CODEC_2X 0 = DISABLE 1 = ENABLE
14	RW	0x1	CLR_DAM2_RST: Clear reset for DAM2 0 = DISABLE 1 = ENABLE
13	RW	0x1	CLR_DAM1_RST: Clear reset for DAM1 0 = DISABLE 1 = ENABLE
12	RW	0x1	CLR_DAM0_RST: Clear reset for DAM0 0 = DISABLE 1 = ENABLE
11	RW	0x1	CLR_APBIF_RST: Clear reset for APBIF 0 = DISABLE 1 = ENABLE
10	RW	0x1	CLR_AUDIO_RST: Clear reset for AUDIO 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
9	RW	0x1	CLR_SPI6_RST: Clear reset for SPI6 0 = DISABLE 1 = ENABLE
8	RW	0x1	CLR_SPI5_RST: Clear reset for SPI5 0 = DISABLE 1 = ENABLE
7	RW	0x1	CLR_I2C4_RST: Clear reset for I2C4 0 = DISABLE 1 = ENABLE
6	RW	0x1	CLR_I2S4_RST: Clear reset for I2S4 0 = DISABLE 1 = ENABLE
5	RW	0x1	CLR_I2S3_RST: Clear reset for I2S3 0 = DISABLE 1 = ENABLE
3	RW	0x1	CLR_MSELECT_RST: Clear reset for MSELECT 0 = DISABLE 1 = ENABLE
1	RO	X	CLR_CPULP_RST: Clear reset for CPULP. [CCLK Multi-Address]. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE
0	RO	X	CLR_CPUG_RST: Clear reset for CPUG. [CCLK Multi-Address]. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE

5.7.147 CLK_RST_CONTROLLER_RST_DEV_W_SET_0

Offset: 0x438 │ Read/Write: R/W │ Reset: 0x1f407fff (0bxxx11111x10xxxxxx111111111111111)

Bit	Reset	Description
28	0x1	SET_XUSB_SS_RST: Set reset for the XUSB_SS logic 0 = DISABLE 1 = ENABLE
27	0x1	SET_DVFS_RST: Set reset for the CLDVFS 0 = DISABLE 1 = ENABLE
26	0x1	SET_ADX0_RST: Set reset for the ADX0 0 = DISABLE 1 = ENABLE
25	0x1	SET_AMX0_RST: Set reset for the AMX0 0 = DISABLE 1 = ENABLE
21	0x0	SET_ENTROPY_RST: Set reset for the ENTROPY logic. 0 = DISABLE 1 = ENABLE
14	0x1	SET_XUSB_PADCTL_RST: Set reset for the XUSB_PADCTL logic. 0 = DISABLE 1 = ENABLE
13	0x1	SET_RESERVED10_RST: Reserved.
12	0x1	SET_RESERVED9_RST: Reserved.
11	0x1	SET_RESERVED8_RST: Reserved.

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

5.7.148 CLK_RST_CONTROLLER_RST_DEV_W_CLR_0

Offset: 0x43c │ Read/Write: R/W │ Reset: 0x1f407fff (0bxxx11111x10xxxxxx111111111111111)

Bit	R/W	Reset	Description
28	RW	0x1	CLR_XUSB_SS_RST: Clear reset for the XUSB_SS logic. 0 = DISABLE 1 = ENABLE
27	RW	0x1	CLR_DVFS_RST: Clear reset for CLDVFS 0 = DISABLE 1 = ENABLE
26	RW	0x1	CLR_ADX0_RST: Clear reset for ADX0 0 = DISABLE 1 = ENABLE
25	RW	0x1	CLR_AMX0_RST: Clear reset for AMX0 0 = DISABLE 1 = ENABLE
21	RW	0x0	CLR_ENTROPY_RST: Clear reset for the ENTROPY logic 0 = DISABLE 1 = ENABLE
14	RW	0x1	CLR_XUSB_PADCTL_RST: Clear reset for XUSB_PADCTL logic 0 = DISABLE 1 = ENABLE
13	RW	0x1	CLR_RESERVED10_RST: Reserved.
12	RW	0x1	CLR_RESERVED9_RST: Reserved.
11	RW	0x1	CLR_RESERVED8_RST: Reserved.
10	RW	0x1	CLR_RESERVED7_RST: Reserved.
9	RW	0x1	CLR_RESERVED6_RST: Reserved.
8	RW	0x1	CLR_CEC_RST: Clear reset for CEC 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
7	RW	0x1	CLR_RESERVED5_RST: Reserved.
6	RW	0x1	CLR_RESERVED4_RST: Reserved.
5	RW	0x1	CLR_RESERVED3_RST: Reserved.
4	RW	0x1	CLR_RESERVED2_RST: Reserved.
3	RW	0x1	CLR_RESERVED1_RST: Reserved.
2	RW	0x1	CLR_RESERVED0_RST: Reserved.
1	RW	0x1	CLR_SATACOLD_RST: Clear reset for SATACOLD 0 = DISABLE 1 = ENABLE
0	RW	0x1	CLR_HDA2HDMICODEC_RST: Clear reset for HDA2HDMICODEC 0 = DISABLE 1 = ENABLE

5.7.149 CLK_RST_CONTROLLER_CLK_ENB_V_SET_0

Offset: 0x440 | Read/Write: R/W | Reset: 0x00400000 (0b0000000001xxxxx00000000000000x00)

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 S/PDIF 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

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
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_SPI6: Set enable clock to SPI6 0 = DISABLE 1 = ENABLE
8	0x0	SET_CLK_ENB_SPI5: Set enable clock to SPI5 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
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

5.7.150 CLK_RST_CONTROLLER_CLK_ENB_V_CLR_0

Offset: 0x444 │ Read/Write: R/W │ Reset: 0x00400000 (0b0000000001xxxxx00000000000000x00)

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
26	0x0	CLR_CLK_ENB_EXTPERIPH3: Clear enable clock to EXTPERIPH3 0 = DISABLE

Bit	Reset	Description
		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 S/PDIF 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_SPI6: Clear enable clock to SPI6 0 = DISABLE 1 = ENABLE
8	0x0	CLR_CLK_ENB_SPI5: Clear enable clock to SPI5 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

Bit	Reset	Description
		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

5.7.151 CLK_RST_CONTROLLER_CLK_ENB_W_SET_0

Offset: 0x448 | Read/Write: R/W | Reset: 0x002000fc (0bxx000000x01000000x00000011111100)

Bit	Reset	Description
29	0x0	SET_CLK_ENB_EMC_DLL: Set enable clock to EMC_DLL 0 = DISABLE 1 = ENABLE
28	0x0	SET_CLK_ENB_XUSB_SS: Set enable clock to XUSB_SS 0 = DISABLE 1 = ENABLE
27	0x0	SET_CLK_ENB_DVFS: Set enable clock to CLDVFS 0 = DISABLE 1 = ENABLE
26	0x0	SET_CLK_ENB_ADX0: Set enable clock to ADX0 0 = DISABLE 1 = ENABLE
25	0x0	SET_CLK_ENB_AMX0: Set the enable clock to AMX0 0 = DISABLE 1 = ENABLE
21	0x1	SET_CLK_ENB_ENTROPY: Set the enable clock to ENTROPY. 0 = DISABLE 1 = ENABLE
20	0x0	SET_CLK_ENB_DSIB_LP: Set the enable clock to DSIB LP. 0 = DISABLE 1 = ENABLE
19	0x0	SET_CLK_ENB_DSIA_LP: Set the enable clock to DSIA LP. 0 = DISABLE 1 = ENABLE
18	0x0	SET_CLK_ENB_CILE: Set the enable clock to CSI CILE. 0 = DISABLE 1 = ENABLE
17	0x0	SET_CLK_ENB_CILCD: Set the enable clock to CSI CILC and CILD. 0 = DISABLE 1 = ENABLE
16	0x0	SET_CLK_ENB_CILAB: Set the enable clock to CSI CILA and CILB. 0 = DISABLE 1 = ENABLE
15	0x0	SET_CLK_ENB_XUSB: Set the enable clock to XUSB 0 = DISABLE 1 = ENABLE
13	0x0	SET_CLK_ENB_MIPI_IOBIST: Set the enable clock to MIPI IOBIST 0 = DISABLE 1 = ENABLE
12	0x0	SET_CLK_ENB_SATA_IOBIST: Set the enable clock to SATA IOBIST 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
11	0x0	SET_CLK_ENB_HDMI_IOBIST: Set the enable clock to HDMI IOBIST 0 = DISABLE 1 = ENABLE
10	0x0	SET_CLK_ENB_EMC_IOBIST: Set the enable clock to EMC IOBIST 0 = DISABLE 1 = ENABLE
9	0x0	SET_CLK_ENB_PCIE2_IOBIST: Set the enable clock to PCIE2 IOBIST 0 = DISABLE 1 = ENABLE
8	0x0	SET_CLK_ENB_CEC: Set the enable clock to CEC 0 = DISABLE 1 = ENABLE
7	0x1	SET_CLK_ENB_PCIERX5: Set the enable clock to PCIERX5 0 = DISABLE 1 = ENABLE
6	0x1	SET_CLK_ENB_PCIERX4: Set the enable clock to PCIERX4 0 = DISABLE 1 = ENABLE
5	0x1	SET_CLK_ENB_PCIERX3: Set the enable clock to PCIERX3 0 = DISABLE 1 = ENABLE
4	0x1	SET_CLK_ENB_PCIERX2: Set the enable clock to PCIERX2 0 = DISABLE 1 = ENABLE
3	0x1	SET_CLK_ENB_PCIERX1: Set the enable clock to PCIERX1 0 = DISABLE 1 = ENABLE
2	0x1	SET_CLK_ENB_PCIERX0: Set the enable clock to PCIERX0 0 = DISABLE 1 = ENABLE
1	0x0	SET_CLK_ENB_RESERVED0: Reserved SATACOLD 0 = DISABLE 1 = ENABLE
0	0x0	SET_CLK_ENB_HDA2HDMICODEC: Set the enable clock to HDA2HDMICODEC 0 = DISABLE 1 = ENABLE

5.7.152 CLK_RST_CONTROLLER_CLK_ENB_W_CLR_0

Offset: 0x44c │ Read/Write: R/W │ Reset: 0x002000fc (0bxx000000x01000000x00000011111100)

Bit	Reset	Description
29	0x0	CLR_CLK_ENB_EMC_LATENCY: Clear the enable clock to EMC_LATENCY 0 = DISABLE 1 = ENABLE
28	0x0	CLR_CLK_ENB_XUSB_SS: Clear the enable clock to XUSB_SS 0 = DISABLE 1 = ENABLE
27	0x0	CLR_CLK_ENB_DVFS: Clear the enable clock to CLDVFS 0 = DISABLE 1 = ENABLE
26	0x0	CLR_CLK_ENB_ADX0: Clear the enable clock to ADX0 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
25	0x0	CLR_CLK_ENB_AMX0: Clear the enable clock to AMX0 0 = DISABLE 1 = ENABLE
21	0x1	CLR_CLK_ENB_ENTROPY: Clear the enable clock to ENTROPY. 0 = DISABLE 1 = ENABLE
20	0x0	CLR_CLK_ENB_DSIB_LP: Clear the enable clock to DSIB LP. 0 = DISABLE 1 = ENABLE
19	0x0	CLR_CLK_ENB_DSIA_LP: Clear the enable clock to DSIA LP. 0 = DISABLE 1 = ENABLE
18	0x0	CLR_CLK_ENB_CILE: Clear the enable clock to CSI CILE. 0 = DISABLE 1 = ENABLE
17	0x0	CLR_CLK_ENB_CILCD: Clear the enable clock to CSI CILC and CILD. 0 = DISABLE 1 = ENABLE
16	0x0	CLR_CLK_ENB_CILAB: Clear the enable clock to CSI CILA and CILB. 0 = DISABLE 1 = ENABLE
15	0x0	CLR_CLK_ENB_XUSB: Clear the enable clock to XUSB 0 = DISABLE 1 = ENABLE
13	0x0	CLR_CLK_ENB_MIPI_IOBIST: Clear the enable clock to MIPI IOBIST 0 = DISABLE 1 = ENABLE
12	0x0	CLR_CLK_ENB_SATA_IOBIST: Clear the enable clock to SATA IOBIST 0 = DISABLE 1 = ENABLE
11	0x0	CLR_CLK_ENB_HDMI_IOBIST: Clear the enable clock to HDMI IOBIST 0 = DISABLE 1 = ENABLE
10	0x0	CLR_CLK_ENB_EMC_IOBIST: Clear the enable clock to EMC IOBIST 0 = DISABLE 1 = ENABLE
9	0x0	CLR_CLK_ENB_PCIE2_IOBIST: Clear the enable clock to PCIE2 IOBIST 0 = DISABLE 1 = ENABLE
8	0x0	CLR_CLK_ENB_CEC: Clear the enable clock to CEC 0 = DISABLE 1 = ENABLE
7	0x1	CLR_CLK_ENB_PCIERX5: Clear the enable clock to PCIERX5 0 = DISABLE 1 = ENABLE
6	0x1	CLR_CLK_ENB_PCIERX4: Clear the enable clock to PCIERX4 0 = DISABLE 1 = ENABLE
5	0x1	CLR_CLK_ENB_PCIERX3: Clear the enable clock to PCIERX3 0 = DISABLE 1 = ENABLE
4	0x1	CLR_CLK_ENB_PCIERX2: Clear the enable clock to PCIERX2 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
3	0x1	CLR_CLK_ENB_PCIERX1: Clear the enable clock to PCIERX1 0 = DISABLE 1 = ENABLE
2	0x1	CLR_CLK_ENB_PCIERX0: Clear the enable clock to PCIERX0 0 = DISABLE 1 = ENABLE
1	0x0	CLR_CLK_ENB_RESERVED0: Reserved.
0	0x0	CLR_CLK_ENB_HDA2HDMICODEC: Clear the enable clock to HDA2HDMICODEC 0 = DISABLE 1 = ENABLE

5.7.153 CLK_RST_CONTROLLER_RST_CPUG_CMPLX_SET_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

Bit	Reset	Description
30	0x0	SET_PRESETDBG: 1 = assert nPRESETDBG to the CoreSight. 0 = DISABLE 1 = ENABLE
29	0x1	SET_NONCPURESET: 1 = Assert reset to the whole nonCPU region of the CPU. For FCCPLEX, by default the NONCPU region is not power-gated but the reset is asserted because CRAIL is power-gated by default. 0 = DISABLE 1 = ENABLE
28	0x0	SET_PERIPHRESET: Reserved.
24	0x0	SET_L2RESET: 1 = Assert nL2RESET to CPU. 0 = DISABLE 1 = ENABLE
23	0x0	SET_CXRESET3: 1 = Assert nCXRESET to CPU3. 0 = DISABLE 1 = ENABLE
22	0x0	SET_CXRESET2: 1 = Assert nCXRESET to CPU2. 0 = DISABLE 1 = ENABLE
21	0x0	SET_CXRESET1: 1 = Assert nCXRESET to CPU1. 0 = DISABLE 1 = ENABLE
20	0x0	SET_CXRESET0: 1 = Assert nCXRESET to CPU0. 0 = DISABLE 1 = ENABLE
19	0x0	SET_CORERESET3: 1 = Assert nCORERESET to CPU3. 0 = DISABLE 1 = ENABLE
18	0x0	SET_CORERESET2: 1 = Assert nCORERESET to CPU2. 0 = DISABLE 1 = ENABLE
17	0x0	SET_CORERESET1: 1 = Assert nCORERESET to CPU1. 0 = DISABLE 1 = ENABLE
16	0x0	SET_CORERESET0: 1 = Assert nCORERESET to CPU0. 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
15	0x1	SET_DBGRESET3: 1 = Assert nDBGRESET to CPU3. 0 = DISABLE 1 = ENABLE
14	0x1	SET_DBGRESET2: 1 = Assert nDBGRESET to CPU2. 0 = DISABLE 1 = ENABLE
13	0x1	SET_DBGRESET1: 1 = Assert nDBGRESET to CPU1. 0 = DISABLE 1 = ENABLE
12	0x1	SET_DBGRESET0: 1 = Assert nDBGRESET to CPU0. 0 = DISABLE 1 = ENABLE
11	0x1	SET_WDRESET3: Reserved.
10	0x1	SET_WDRESET2: Reserved.
9	0x1	SET_WDRESET1: Reserved.
8	0x0	SET_WDRESET0: Reserved.
7	0x1	SET_DERESET3: Reserved.
6	0x1	SET_DERESET2: Reserved.
5	0x1	SET_DERESET1: Reserved.
4	0x0	SET_DERESET0: Reserved.
3	0x1	SET_CPURESET3: 1 = Assert nCPUPORRESET to CPU3. 0 = DISABLE 1 = ENABLE
2	0x1	SET_CPURESET2: 1 = Assert nCPUPORRESET to CPU2. 0 = DISABLE 1 = ENABLE
1	0x1	SET_CPURESET1: 1 = Assert nCPUPORRESET to CPU1. 0 = DISABLE 1 = ENABLE
0	0x1	SET_CPURESET0: 1 = Assert nCPUPORRESET to CPU0. 0 = DISABLE 1 = ENABLE

5.7.154 CLK_RST_CONTROLLER_RST_CPUG_CMPLX_CLR_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

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

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
23	0x0	CLR_CXRESET3: 1 = De-assert nCXRESET to CPU3. 0 = DISABLE 1 = ENABLE
22	0x0	CLR_CXRESET2: 1 = De-assert nCXRESET to CPU2. 0 = DISABLE 1 = ENABLE
21	0x0	CLR_CXRESET1: 1 = De-assert nCXRESET to CPU1. 0 = DISABLE 1 = ENABLE
20	0x0	CLR_CXRESET0: 1 = De-assert nCXRESET to CPU0. 0 = DISABLE 1 = ENABLE
19	0x0	CLR_CORERESET3: 1 = De-assert nCORERESET to CPU3. 0 = DISABLE 1 = ENABLE
18	0x0	CLR_CORERESET2: 1 = De-assert nCORERESET to CPU2. 0 = DISABLE 1 = ENABLE
17	0x0	CLR_CORERESET1: 1 = De-assert nCORERESET to CPU1. 0 = DISABLE 1 = ENABLE
16	0x0	CLR_CORERESET0: 1 = De-assert nCORERESET to CPU0. 0 = DISABLE 1 = ENABLE
15	0x1	CLR_DBGRESET3: 1 = De-assert nDBGRESET to CPU3. 0 = DISABLE 1 = ENABLE
14	0x1	CLR_DBGRESET2: 1 = De-assert nDBGRESET to CPU2. 0 = DISABLE 1 = ENABLE
13	0x1	CLR_DBGRESET1: 1 = De-assert nDBGRESET to CPU1. 0 = DISABLE 1 = ENABLE
12	0x1	CLR_DBGRESET0: 1 = De-assert nDBGRESET to CPU0. 0 = DISABLE 1 = ENABLE
11	0x1	CLR_WDRESET3: Reserved.
10	0x1	CLR_WDRESET2: Reserved.
9	0x1	CLR_WDRESET1: Reserved.
8	0x0	CLR_WDRESET0: Reserved.
7	0x1	CLR_DERESET3: Reserved.
6	0x1	CLR_DERESET2: Reserved.
5	0x1	CLR_DERESET1: Reserved.
4	0x0	CLR_DERESET0: Reserved.
3	0x1	CLR_CPURESET3: 1 = De-assert nCPUPORRESET to CPU3. 0 = DISABLE 1 = ENABLE
2	0x1	CLR_CPURESET2: 1 = De-assert nCPUPORRESET to CPU2. 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
1	0x1	CLR_CPURESET1: 1 = De-assert nCPUPORRESET to CPU1. 0 = DISABLE 1 = ENABLE
0	0x1	CLR_CPURESET0: 1 = De-assert nCPUPORRESET to CPU0. 0 = DISABLE 1 = ENABLE

5.7.155 CLK_RST_CONTROLLER_RST_CPULP_CMPLX_SET_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

Offset: 0x458 │ Read/Write: R/W │ Reset: 0x20001001 (0bx010xxx0xxx0xxx0xxx1xxx0xxx0xxx1)

Bit	Reset	Description
30	0x0	SET_PRESETDBG: 1 = assert nPRESETDBG to the CoreSight. 0 = DISABLE 1 = ENABLE
29	0x1	SET_NONCPURESET: 1 = Assert reset to the whole nonCPU region of the CPU. 0 = DISABLE 1 = ENABLE
28	0x0	SET_PERIPHRESET: Reserved
24	0x0	SET_L2RESET: 1 = Assert nL2RESET to the CPU. 0 = DISABLE 1 = ENABLE
20	0x0	SET_CXRESET0: 1 = Assert nCXRESET to CPU0. 0 = DISABLE 1 = ENABLE
16	0x0	SET_CORERESET0: 1 = Assert nCORERESET to CPU0. 0 = DISABLE 1 = ENABLE
12	0x1	SET_DBGRESET0: 1 = Assert nDBGRESET to CPU0. 0 = DISABLE 1 = ENABLE
8	0x0	SET_WDRESET0: Reserved.
4	0x0	SET_DERESET0: Reserved.
1	0x1	SET_CPURESET1: N/A
0	0x1	SET_CPURESET0: 1 = assert nCPUPORRESET to CPU0. 0 = DISABLE 1 = ENABLE

5.7.156 CLK_RST_CONTROLLER_RST_CPULP_CMPLX_CLR_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

Offset: 0x45c │ Read/Write: R/W │ Reset: 0x20001001 (0bx010xxx0xxx0xxx0xxx1xxx0xxx0xxx1)

Bit	Reset	Description
30	0x0	CLR_PRESETDBG: 1 = De-assert nPRESETDBG to the CoreSight. 0 = DISABLE 1 = ENABLE
29	0x1	CLR_NONCPURESET: 1 = De-assert reset to the whole nonCPU region of the CPU. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
28	0x0	CLR_PERIPHRESET: Reserved.
24	0x0	CLR_L2RESET: 1 = De-assert nL2RESET to CPU. 0 = DISABLE 1 = ENABLE
20	0x0	CLR_CXRESET0: 1 = De-assert nCXRESET to CPU0. 0 = DISABLE 1 = ENABLE
16	0x0	CLR_CORERESET0: 1 = De-assert nCORERESET to CPU0. 0 = DISABLE 1 = ENABLE
12	0x1	CLR_DBGRESET0: 1 = De-assert nDBGRESET to CPU0. 0 = DISABLE 1 = ENABLE
8	0x0	CLR_WDRESET0: Reserved.
4	0x0	CLR_DERESET0: Reserved.
0	0x1	CLR_CPURESET0: 1 = De-assert nCPUPORRESET to CPU0. 0 = DISABLE 1 = ENABLE

5.7.157 CLK_RST_CONTROLLER_CLK_CPUG_CMPLX_SET_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

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

5.7.158 CLK_RST_CONTROLLER_CLK_CPUG_CMPLX_CLR_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

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

Bit	Reset	Description
		1 = ENABLE
8	0x0	CLR_CPU0_CLK_STP: 1 = De-assert CPU0 clock stop 0 = DISABLE 1 = ENABLE

5.7.159 CLK_RST_CONTROLLER_CLK_CPULP_CMPLX_SET_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

Offset: 0x468 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx)

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

5.7.160 CLK_RST_CONTROLLER_CLK_CPULP_CMPLX_CLR_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

Offset: 0x46c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx)

Bit	Reset	Description
8	0x0	CLR_CPU0_CLK_STP: 1 = De-assert CPU0 clock stop 0 = DISABLE 1 = ENABLE

5.7.161 CLK_RST_CONTROLLER_CPU_CMPLX_STATUS_0

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

Bit	Reset	Description
30	X	PRESETDBG: ENABLE = nPRESETDBG asserted to the CoreSight. 0 = DISABLE 1 = ENABLE
29	X	NONCPURESET: ENABLE = Reset asserted to the whole nonCPU region of the CPU 0 = DISABLE 1 = ENABLE
28	X	MCRESET: ENABLE = mcreset_ asserted 0 = DISABLE 1 = ENABLE
27	X	CSITEPTMRESET: ENABLE = nCSITEPTMRESET asserted 0 = DISABLE 1 = ENABLE
26	X	AXICIFRESET: ENABLE = nAXICIFRESET asserted 0 = DISABLE 1 = ENABLE
25	X	MPSELECTRESET: ENABLE = mselectreset asserted 0 = DISABLE 1 = ENABLE
24	X	L2RESET: ENABLE = L2RESET asserted to the CPU 0 = DISABLE 1 = ENABLE
23	X	CXRESET3: ENABLE = nCXRESET asserted to CPU3 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
22	X	CXRESET2: ENABLE = nCXRESET asserted to CPU2 0 = DISABLE 1 = ENABLE
21	X	CXRESET1: ENABLE = nCXRESET asserted to CPU1 0 = DISABLE 1 = ENABLE
20	X	CXRESET0: ENABLE = nCXRESET asserted to CPU0 0 = DISABLE 1 = ENABLE
19	X	CORERESET3: ENABLE = nCORERESET asserted to CPU3 0 = DISABLE 1 = ENABLE
18	X	CORERESET2: ENABLE = nCORERESET asserted to CPU2 0 = DISABLE 1 = ENABLE
17	X	CORERESET1: ENABLE = nCORERESET asserted to CPU1 0 = DISABLE 1 = ENABLE
16	X	CORERESET0: ENABLE = nCORERESET asserted to CPU0 0 = DISABLE 1 = ENABLE
15	X	DBGRESET3: ENABLE = nDBGRESET asserted to CPU3 0 = DISABLE 1 = ENABLE
14	X	DBGRESET2: ENABLE = nDBGRESET asserted to CPU2 0 = DISABLE 1 = ENABLE
13	X	DBGRESET1: ENABLE = nDBGRESET asserted to CPU1 0 = DISABLE 1 = ENABLE
12	X	DBGRESET0: ENABLE = nDBGRESET asserted to CPU0 0 = DISABLE 1 = ENABLE
11	X	WDRESET3: Reserved.
10	X	WDRESET2: Reserved.
9	X	WDRESET1: Reserved.
8	X	WDRESET0: Reserved.
3	X	CPURESET3: ENABLE = nCPUPORRESET asserted to CPU3 0 = DISABLE 1 = ENABLE
2	X	CPURESET2: ENABLE = nCPUPORRESET asserted to CPU2 0 = DISABLE 1 = ENABLE
1	X	CPURESET1: ENABLE = nCPUPORRESET asserted to CPU1 0 = DISABLE 1 = ENABLE
0	X	CPURESET0: ENABLE = nCPUPORRESET asserted to CPU0 0 = DISABLE 1 = ENABLE

5.7.162 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: 0x478 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

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

5.7.163 CLK_RST_CONTROLLER_INTMASK_0

Interrupt Mask Register.

Offset: 0x47c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

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

5.7.164 CLK_RST_CONTROLLER_UTMIP_PLL_CFG0_0

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

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

UTMIP PLL Configuration Register 0

Offset: 0x480 │ Read/Write: R/W │ Reset: 0x005001XX (0bx00000000101000000000001xxxxx000)

Bit	R/W	Reset	Description
30:29	RW	0x0	UTMIP_PLL_VREG_BG_CTRL: Voltage regulator and band-gap probe mux control. VREG_BG_CTRL = 0 => VREG_BG_PRB = 0. VREG_BG_CTRL = 1 => VREG_BG_PRB = bandgap. VREG_BG_CTRL = 2 => VREG_BG_PRB = Vregulator. See cell specification.
28:27	RW	0x0	UTMIP_PLL_VREG_CTRL: Voltage regulator voltage level control. See cell specification.
26:25	RW	0x0	UTMIP_PLL_KCP: KCP of the UTMIP PHY PLL. Charge Pump Gain control. Default value is zero. See cell specification.
23:16	RW	0x50	UTMIP_PLL_NDIV: NDIV[7:0] input of UTMIP PLL. This is the feedback divider on the VCO feedback. 0x0 is not allowed. See cell specification.
15:8	RW	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.
3	RO	X	UTMIP_PLL_RESERVED: Reserved.
2	RW	0x0	UTMIP_PLL_LOCK_OVR: LOCK_OVERRIDE control of the UTMIP PLL. Forces PLL_LOCK to 1. See cell specification.
1	RW	0x0	UTMIP_PLL_EN_FSTLCK: EN_FSTLCK input of UTMIP PLL. Reserving pin for PLL fast lock circuitry. See cell specification.
0	RW	0x0	UTMIP_PLL_ENB_LCKDET: ENB_LCKDET input of UTMIP PLL. When set to 1, powers down the lock detect. Setting ENB_LCKDET=X, LOCK_OVERRIDE=1, PLL_LOCK=1, and PLL_FREQLOCK=1 also powers down lock detect. 0 0 0->1(after lock) 0->1 (after lock) 1 0 0 0 lock detect power down. See cell specification.

5.7.165 CLK_RST_CONTROLLER_UTMIP_PLL_CFG1_0

UTMIP PLL Configuration and Parameters

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

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

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

- PllUOnState: start pllu_enable_count and pll_lock_count
  - Wait ~1 µs to enable the PLL_U (pllu_enable_count == ClkXtal * PLLU_ENABLE_DLY_COUNT * 8)
  - Wait ~1 ms until PLL_U is stable (pll_lock_count == ClkXtal * PLLU_STABLE_COUNT * 256) => USB_PHY PLL_ENABLE
  - pll_active_count = 0
    - Wait 5 µs to enable pll_active: pll_active_count == ClkXtal * PLL_ACTIVE_DLY_COUNT * 16 => USB_PHY PLL_ACTIVE
  - Wait ~2.5 ms until USB_PHY is stable (pll_lock_count == ClkXtal * XTAL_FREQ_COUNT * 256) => USB_PHY

Values for a 19.2 MHz Xtal clock are:

- PLLU_ENABLE_DLY_COUNT[4:0] = 1 * 12 / 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: 0x484 | Read/Write: R/W | Reset: 0x180150c0 (0b00011000000000001010100000011000000)

Bit	Reset	Description
31:27	0x3	UTMIP_PLLU_ENABLE_DLY_COUNT: Controls the wait time to enable PLL_U when coming out of suspend or reset.
26:18	0x0	UTMIP_PLL_RSVD: Reserved
17	0x0	UTMIP_FORCE_PLLU_POWERUP: Force PLL_U into power up.
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: Obsolete.
12	0x1	UTMIP_FORCE_PLL_ACTIVE_POWERDOWN: Obsolete.
11:0	0xc0	UTMIP_XTAL_FREQ_COUNT: Determines the time to wait until the output of UTMIP PLL is considered stable.

5.7.166 CLK_RST_CONTROLLER_UTMIP_PLL_CFG2_0

UTMIP Miscellaneous Configurations

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

Bit	R/W	Reset	Description
30	RW	0x1	UTMIP_PHY_XTAL_CLOCKEN: Selects whether to enable the crystal clock in the module.
29:26	RO	X	UTMIP_FORCE_PD_RESERVED: Reserved.
25	RW	0x0	UTMIP_FORCE_PD_SAMP_D_POWERUP: Force UTMIP PLL PD_SAMP_D input (XUSB DEV) into power up.
24	RW	0x1	UTMIP_FORCE_PD_SAMP_D_POWERDOWN: Force UTMIP PLL PD_SAMP_D input (XUSB DEV) into power down. (Overrides FORCE_PD_SAMP_D_POWERUP.)
23:18	RW	0xc	UTMIP_PLL_ACTIVE_DLY_COUNT: 10 us / (1/19.2MHz) = 192 / 16 = 12
17:6	RW	0x600	UTMIP_PLLU_STABLE_COUNT: PLLU frequency lock delay (See comments above on correct delay and calculation)
5	RW	0x0	UTMIP_FORCE_PD_SAMP_C_POWERUP: Force UTMIP PLL PD_SAMP_C input into power up.
4	RW	0x1	UTMIP_FORCE_PD_SAMP_C_POWERDOWN: Force UTMIP PLL PD_SAMP_C input into power down. (Overrides FORCE_PD_SAMP_C_POWERUP.)
3	RW	0x0	UTMIP_FORCE_PD_SAMP_B_POWERUP: Force UTMIP PLL PD_SAMP_B input (XUSB HOST) into power up.
2	RW	0x1	UTMIP_FORCE_PD_SAMP_B_POWERDOWN: Force UTMIP PLL PD_SAMP_B input (XUSB HOST) into power down. (Overrides FORCE_PD_SAMP_B_POWERUP.)
1	RW	0x0	UTMIP_FORCE_PD_SAMP_A_POWERUP: Force UTMIP PLL PD_SAMP_A input into power up.
0	RW	0x1	UTMIP_FORCE_PD_SAMP_A_POWERDOWN: Force UTMIP PLL PD_SAMP_A input into power down. (Overrides FORCE_PD_SAMP_A_POWERUP.)

5.7.167 CLK_RST_CONTROLLER_PLLE_AUX_0

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

Offset: 0x48c │ Read/Write: R/W │ Reset: 0x0X044070 (0b0xx0xx10000001000100000001110000)

Bit	R/W	Reset	Description
31	RW	SKIP	PLLE_SS_SEQ_INCLUDE: 0=Skip, 1=Include. Include PLLE spread spectrum power sequencer. If this bit needs to be toggled, do it before writing '1' to PLLE_SEQ_ENABLE. 0 = SKIP 1 = INCLUDE
28	RW	0x0	PLLE_REF_SEL_PLLREFE: PLLE input reference clock source select2. 0 = Select the setting of the PLLE_REF_SRC field mentioned below, 1 = Select PLLREFE_out which is 600M (use pre-divM=50 for 12 MHz reference)
27:26	RO	X	PLLE_SEQ_STATE: PLLE power sequencer state 0 = SEQ_OFF 1 = SEQ_ON 2 = SEQ_BUSY
25	RW	SEQ_ON	PLLE_SEQ_START_STATE: PLLE power sequencer start state 0 = SEQ_OFF 1 = SEQ_ON
24	RW	DISABLE	PLLE_SEQ_ENABLE: PLLE power sequencer enable. Start state is loaded on first cycle after set. 0 = DISABLE 1 = ENABLE
23:16	RW	0x4	PLLE_SS_DLY: PT6: Delay between spread spectrum ON<->OFF transitions in SS power toggle sequence: ON->OFF->ON. Range is 0-255 ms in 1 ms steps.
15:8	RW	0x40	PLLE_LOCK_DLY: PT5: Delay from PLLE ENABLE to lock. Range is 0-128 µs in 1 µs steps
7	RW	0x0	TEST_FAST_PT: 0=Normal timer steps. 1=Fast timer steps. Fast programmable timer steps size for testing. Normal is per µs or ms, fast is per oscillator clock. Applies to all programmable timer counters in SATA, PCIe and PLLE seq.
6	RW	0x1	PLLE_SS_SWCTL: 0=PLLE spread spectrum configuration setup by hardware, 1=Setup by software during training
5	RW	0x1	PLLE_CONFIG_SWCTL: This bit should never be written to 0. Do not change the setup bits once written by software. 0=PLLE config setup by hardware, 1=PLLE setup by software during training. Affects resettable bits when PLL is off (SETUP and EXT_SETUP bits). All other config bits are left untouched as initialized, for example, M/N/P, SS coefficient.
4	RW	0x1	PLLE_ENABLE_SWCTL: 0=PLLE enable by hardware, 1=PLLE enable by software
3	RW	0x0	PLLE_USE_LOCKDET: 0=Use programmable delay, 1=use lockdet signals from PLLE
2	RW	OSC_DIV	PLLE_REF_SRC: PLLE input reference clock source select. 0=OSC_DIV (options: mux (PLLs, ext_osc), /2, /4), 1=PLLP_OUT0 (use pre-divM=18 for 12 MHz reference) 0 = OSC_DIV 1 = PLLP_OUT0
1	RW	DISABLE	PLLE_CML1_OEN: PLLE CML1 clock out enable. Used for SATA. 0 = DISABLE 1 = ENABLE
0	RW	DISABLE	PLLE_CML0_OEN: PLLE CML0 clock out enable. Used for PCIe. 0 = DISABLE 1 = ENABLE

5.7.168 CLK_RST_CONTROLLER_SATA_PLL_CFG0_0

SATA Power Sequencer Input Software Control Program Guide

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

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

Offset: 0x490 │ Read/Write: R/W │ Reset: 0x0X000003 (0bxxxxxx10xxxxxxxxxxxxxxxx0000x011)

Bit	R/W	Reset	Description
27:26	RO	X	SATA_SEQ_STATE: SATA power sequencer state 0 = SEQ_OFF 1 = SEQ_ON 2 = SEQ_BUSY
25	RW	SEQ_ON	SATA_SEQ_START_STATE: SATA power sequencer start state 0 = SEQ_OFF 1 = SEQ_ON
24	RW	DISABLE	SATA_SEQ_ENABLE: SATA power sequencer enable. Start state is loaded on first cycle after set. 0 = DISABLE 1 = ENABLE
7	RW	0x0	SATA_SEQ_PADPLL_PD_INPUT_VALUE: 0=Pad PLL is powered up, 1=Software control pad PLL powered down (PD) by setting this and SATA_SEQ_IN_SWCTL bit
6	RW	0x0	SATA_SEQ_LANE_PD_INPUT_VALUE: 0=IO PHY is powered up, 1=Software control I/O PHY powered down by setting this and SATA_SEQ_IN_SWCTL bit
5	RW	0x0	SATA_SEQ_RESET_INPUT_VALUE: 0=SATA reset de-asserted, 1=Software control reset by setting this and SATA_SEQ_IN_SWCTL bit
4	RW	0x0	SATA_SEQ_IN_SWCTL: 0=Seq input by hardware, 1=Seq input by software
2	RW	0x0	SATA_PADPLL_USE_LOCKDET: 0=Use programmable delay, 1=Use lockdet signals from PLL
1	RW	0x1	SATA_PADPLL_RESET_OVERRIDE_VALUE: 0=Pad PLL is powered up, 1=Software control reset by setting this and SATA_PADPLL_RESET_SWCTL bit
0	RW	0x1	SATA_PADPLL_RESET_SWCTL: 0=Reset by hardware, 1=Reset by software

5.7.169 CLK_RST_CONTROLLER_SATA_PLL_CFG1_0

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

Offset: 0x494 │ Read/Write: R/W │ Reset: 0x20202008 (0b00100000001000000010000000001000)

Bit	Reset	Description
31:24	0x20	SATA_LANE_IDDQ2_PADPLL_RESET_DLY: PT4: Delay from placing lane out of IDDQ to PAD PLL in reset. Range is 0-200 µs.
23:16	0x20	SATA_PADPLL_IDDQ2LANE_SLUMBER_DLY: PT3: Delay from SATA PAD PLL out of IDDQ to lane placed out of IDDQ. Range is 0-64 µs.
15:8	0x20	SATA_PADPLL_PU_POST_DLY: PT2: Delay from RESET to SATA PAD PLL lock. Range is 0-64 µs.
7:0	0x8	SATA_LANE_IDDQ2_PADPLL_IDDQ_DLY: PT1: Delay from placing lane in IDDQ to PAD PLL in IDDQ. Range is 0-16 µs.

5.7.170 CLK_RST_CONTROLLER_PCIE_PLL_CFG_0

PCIe Power Sequencer Input Software Control Program Guide

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

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

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

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 de-asserted, 1=Software control reset by setting this and PCIE_SEQ_IN_SWCTL bit
4	RW	0x0	PCIE_SEQ_IN_SWCTL: 0=Sequencer input by hardware, 1=Sequencer input by software
3	RW	ENABLE	PCIE_XCLK_ENABLE_OVERRIDE_VALUE: Override value used only when PCIE_XCLK_ENABLE_SWCTL is set 0 = DISABLE 1 = ENABLE
2	RW	0x1	PCIE_XCLK_ENABLE_SWCTL: 0=XCLK enabled by hardware, 1=XCLK enabled by software

5.7.171 CLK_RST_CONTROLLER_PROG_AUDIO_DLY_CLK_0

- Clock doubler with 1X/2X is available for the S/PDIF clock.
- The CLK_ENB_SPDIF_DOUBLE bits in the CLK_OUT_ENB_V register must be enabled for either SYNC_1X_CLK or SYNC_2X_CLK to operate.
- PROG_DLY_CLK_SPDIF provides programmable delay for the S/DIF clock doublers.

Enable Sync Clocks

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

- CG_1. AUDIO_SYNC_CLK_SPDIF-SYNC_CLK_DIS: disable output of sync_clk mux
- CG_2. CLK_OUT_ENB_V-CLK_ENB_SPDIF_DOUBLER: disable input to doubler
- SEL_3. PROG_AUDIO_DLY_CLK-SPDIF_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_SPDIF_DOUBLER	SPDIF_1X_SEL	SYNC_CLK
0	1	0	2x (Default)
0	1	1	1x
1	1	X	0 (off, recommended)
1	X	X	0 (off)
X	0	X	0 (off)

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

Bit	Reset	Description
29	0x0	SPDIF_1X_SEL: Select S/PDIF 1x sync_clk. 0=sync_2x_clk, 1=sync_clk
23:20	0x7	SYNC_CLK_SPDIF_DELCLK_SEL: 16 taps of selectable delay for SYNC_CLK clock doubler

5.7.172 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S0_0

Audio Sync Clock Source Select

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

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable Audio sync clock.
3:0	0x0	SYNC_CLK_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S0 bit clock. 0010 = I2S1 bit clock. 0011 = I2S2 bit clock. 0100 = I2S3 bit clock. 0101 = I2S4 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S0 2 = I2S1 3 = I2S2 4 = I2S3 5 = I2S4 6 = PLLA_OUT0 7 = EXT_VIMCLK

5.7.173 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S1_0

Audio Sync Clock Source Select

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

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable Audio sync clock
3:0	0x0	SYNC_CLK_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S0 bit clock. 0010 = I2S1 bit clock. 0011 = I2S2 bit clock. 0100 = I2S3 bit clock. 0101 = I2S4 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = reserved 0 = SPDIFIN 1 = I2S0 2 = I2S1 3 = I2S2 4 = I2S3 5 = I2S4 6 = PLLA_OUT0 7 = EXT_VIMCLK

5.7.174 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S2_0

Audio Sync Clock Source Select

Offset: 0x4a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable Audio sync clock
3:0	0x0	SYNC_CLK_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S0 bit clock. 0010 = I2S1 bit clock. 0011 = I2S2 bit clock. 0100 = I2S3 bit clock. 0101 = I2S4 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S0 2 = I2S1 3 = I2S2 4 = I2S3 5 = I2S4 6 = PLLA_OUT0 7 = EXT_VIMCLK

5.7.175 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S3_0

Audio Sync Clock Source Select

Offset: 0x4ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable Audio sync clock
3:0	0x0	SYNC_CLK_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S0 bit clock. 0010 = I2S1 bit clock. 0011 = I2S2 bit clock. 0100 = I2S3 bit clock. 0101 = I2S4 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S0 2 = I2S1 3 = I2S2 4 = I2S3 5 = I2S4 6 = PLLA_OUT0 7 = EXT_VIMCLK

5.7.176 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S4_0

Audio Sync Clock Source Select

Offset: 0x4b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable Audio sync clock
3:0	0x0	SYNC_CLK_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S0 bit clock. 0010 = I2S1 bit clock. 0011 = I2S2 bit clock. 0100 = I2S3 bit clock. 0101 = I2S4 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S0 2 = I2S1 3 = I2S2 4 = I2S3 5 = I2S4 6 = PLLA_OUT0 7 = EXT_VIMCLK

5.7.177 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_SPDIF_0

Audio Sync Clock Source Select

Offset: 0x4b4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable Audio sync clock
3:0	0x0	SYNC_CLK_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S0 bit clock. 0010 = I2S1 bit clock. 0011 = I2S2 bit clock. 0100 = I2S3 bit clock. 0101 = I2S4 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S0 2 = I2S1 3 = I2S2 4 = I2S3 5 = I2S4 6 = PLLA_OUT0 7 = EXT_VIMCLK

5.7.178 CLK_RST_CONTROLLER_PLLD2_BASE_0

Offset: 0x4b8 | Read/Write: R/W | Reset: 0xXX08010c (0b000xx00000001xx00000000100001100)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLD2_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLD2_ENABLE: 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLD2_REF_DIS: 0 = enable reference clock, 1 = disable reference clock. 0 = REF_ENABLE 1 = REF_DISABLE
28	RO	X	PLLD2_FREQLOCK: 0 = not lock, 1 = lock frequency
27	RO	X	PLLD2_LOCK: 0 = not lock, 1 = lock.
26:25	RW	0x0	PLLD2_REF_SRC_SEL: Reference source select sel[0]=0 && sel[1]=1 => ref_src = pllP_out0 sel[0]=0 && sel[1]=0 => ref_src = osc_div_clk 0 = PLL_D 1 = PLL_D2
24	RW	0x0	PLLD2_LOCK_OVERRIDE:Forces PLL_LOCK to 1.
23:20	RW	0x0	PLLD2_PLDIV: 0 = PL divider.
19	RW	0x1	PLLD2_IDDQ: 0 The PLL is powered up 1: Software can put the PLL in IDDQ by setting this bit 0 = OFF 1 = ON
16	RW	0x0	PLLD2_PTS: Base PLLD2 test output select.
15:8	RW	0x1	PLLD2_NDIV: N divider.
7:0	RW	0xc	PLLD2_MDIV: M divider.

5.7.179 CLK_RST_CONTROLLER_PLLD2_MISC_0

Offset: 0x4bc | Read/Write: R/W | Reset: 0xXX000000 (0b00xxx00000000000000000000000000)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLD2_ EN_FSTLCK: Enables Fast lock 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLD2_ LOCK_ENABLE: 0 = Disable, 1 = Enable. 0 = DISABLE 1 = ENABLE
29:27	RO	X	PLLD2_MON_TEST_OUT
26:25	RW	0x0	PLLD2_ KCP:Charge pump gain control.
24	RW	0x0	PLLD2_ KVCO:VCO gain.
23:0	RW	0x0	PLLD2_ SETUP: setup[23:0].

5.7.180 CLK_RST_CONTROLLER_UTMIP_PLL_CFG3_0

UTMIP_PLL_CFG3_0

Offset: 0x4c0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:0	0x0	UTMIP_PLL_SETUP: Debug control bits. Default is 0. setup[6:0]: Lock Detect Controls setup[8:7]: Current source control setup [9]: Forces loop filter to VDDA/2 setup [10]: Corevdd detection override setup [11:20] TBD setup [23:21]: Phase selection on CLKOUTMUX60

5.7.181 CLK_RST_CONTROLLER_PLLREFE_BASE_0

Offset: 0x4c4 │ Read/Write: R/W │ Reset: 0x00000000 (0b000000xxxxxx00000000000000000000)

Bit	Reset	Description
31	DISABLE	PLLREFE_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	DISABLE	PLLREFE_ENABLE: 0 = DISABLE 1 = ENABLE
29	REF_ENABLE	PLLREFE_REF_DIS: 0 = enable reference clock, 1 = disable reference clock. 0 = REF_ENABLE 1 = REF_DISABLE
28:27	0x0	PLLREFE_KCP
26	0x0	PLLREFE_KVCO
19:16	0x0	PLLREFE_PLDIV
15:8	0x0	PLLREFE_NDIV
7:0	0x0	PLLREFE_MDIV

5.7.182 CLK_RST_CONTROLLER_PLLREFE_MISC_0

Offset: 0x4c8 | Read/Write: R/W | Reset: 0x0X010000 (0b000xxxxxxxxx0010000000000000000)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLREFE_EN_FSTLCK: 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLREFE_LOCK_ENABLE: 0 = DISABLE 1 = ENABLE
29	RW	0x0	PLLREFE_LOCK_OVERRIDE. Forces PLLREFE_LOCK and PLLREFE_FREQLOCK to 1.
25	RO	X	PLLREFE_FREQLOCK: 0 = frequency acquisition not achieved, 1 = frequency acquisition occurred. PLLREFE_LOCK_ENABLE must be enabled.
24	RO	X	PLLREFE_LOCK: 0 = not lock (phase+frequency), 1 = lock (phase+frequency). PLLREFE_LOCK_ENABLE must be enabled.
18:17	RW	0x0	PLLREFE_PTS: 00 = PTO is 0 (DISABLE); 01 = PTO is FO; 10 = PTO is VCO out; 11 = Reserved.
16	RW	0x1	PLLREFE_IDDQ: 0 = The PLLREFE is powered up. 1 = Software can put the PLLREFE in IDDQ by setting this bit. 0 = OFF 1 = ON
15:0	RW	0x0	PLLREFE_SETUP

5.7.183 CLK_RST_CONTROLLER_CPU_FINETRIM_BYP_0

CPU Trimmer Registers

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

These registers control the delay through the programmable trimmers (CORE_CLOCKS_shaper) inside the Cortex®-A15 partitions.

- CPU_FINETRIM_BYP
- CPU_FINETRIM_SELECT
- CPU_FINETRIM_DR
- CPU_FINETRIM_DF

- CPU_FINETRIM_F
- CPU_FINETRIM_R

The naming convention of the fields of these registers is:

- FCPU_n: partition n in fast CPU cluster
- SCPU_n: partition n in slow CPU cluster

The partitions related to these registers are as follows:

- FCPU_1: fcpu0 partition
- FCPU_2: fcpu1 partition
- FCPU_3: fcpu2 partition
- FCPU_4: fcpu3 partition
- FCPU_5: fl2 partition
- FCPU_6: ftop partition
- SCPU_7: not used
- SCPU_8: not used
- SCPU_9: not used

Offset: 0x4d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8	0x0	SCPU_9
7	0x0	SCPU_8
6	0x0	SCPU_7
5	0x0	FCPU_6
4	0x0	FCPU_5
3	0x0	FCPU_4
2	0x0	FCPU_3
1	0x0	FCPU_2
0	0x0	FCPU_1

5.7.184 CLK_RST_CONTROLLER_CPU_FINETRIM_SELECT_0

Offset: 0x4d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8	0x0	SCPU_9
7	0x0	SCPU_8
6	0x0	SCPU_7
5	0x0	FCPU_6
4	0x0	FCPU_5
3	0x0	FCPU_4
2	0x0	FCPU_3
1	0x0	FCPU_2
0	0x0	FCPU_1

5.7.185 CLK_RST_CONTROLLER_CPU_FINETRIM_DR_0

Offset: 0x4d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8	0x0	SCPU_9
7	0x0	SCPU_8
6	0x0	SCPU_7
5	0x0	FCPU_6
4	0x0	FCPU_5
3	0x0	FCPU_4
2	0x0	FCPU_3
1	0x0	FCPU_2
0	0x0	FCPU_1

5.7.186 CLK_RST_CONTROLLER_CPU_FINETRIM_DF_0

Offset: 0x4dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8	0x0	SCPU_9
7	0x0	SCPU_8
6	0x0	SCPU_7
5	0x0	FCPU_6
4	0x0	FCPU_5
3	0x0	FCPU_4
2	0x0	FCPU_3
1	0x0	FCPU_2
0	0x0	FCPU_1

5.7.187 CLK_RST_CONTROLLER_CPU_FINETRIM_F_0

Offset: 0x4e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx000000000000000000)

Bit	Reset	Description
17:16	0x0	SCPU_9
15:14	0x0	SCPU_8
13:12	0x0	SCPU_7
11:10	0x0	FCPU_6
9:8	0x0	FCPU_5
7:6	0x0	FCPU_4
5:4	0x0	FCPU_3
3:2	0x0	FCPU_2
1:0	0x0	FCPU_1

5.7.188 CLK_RST_CONTROLLER_CPU_FINETRIM_R_0

Offset: 0x4e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx00000000000000000)

Bit	Reset	Description
17:16	0x0	SCPU_9
15:14	0x0	SCPU_8
13:12	0x0	SCPU_7
11:10	0x0	FCPU_6
9:8	0x0	FCPU_5
7:6	0x0	FCPU_4
5:4	0x0	FCPU_3
3:2	0x0	FCPU_2
1:0	0x0	FCPU_1

5.7.189 CLK_RST_CONTROLLER_PLLC2_BASE_0

There are special startup and programming considerations for this digital PLL. Consult the datasheet for details

Some field descriptions will have a LATCHED or a DIRECT keyword:

- LATCHED: The writes to this field are latched by STROBE. Software should note that while readback will tell the last written value, that might not be the latched value.
- DIRECT: It is a DIRECT read/write field. The writes to this field are not latched by STROBE. So, readback will always tell the correct state.

Offset: 0x4e8 │ Read/Write: R/W │ Reset: 0x0X000101 (0b000xxxxxx000xxxx00000001xxxxxx01)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLC2_BYPASS: DIRECT. 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLC2_ENABLE: DIRECT. 0 = disable, 1 = enable. Will invert and connect to IDDQ 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLC2_REF_DIS: 0 = enable reference clock, 1 = disable reference clock. To go into the minimum power mode for the PLL, this bit must be asserted. 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLC2_FREQ_LOCK: 0 = not lock, 1 = frequency is locked.
26	RO	X	PLLC2_PHASE_LOCK: 0 = not lock, 1 = phase is locked.
22:20	RW	0x0	PLLC2_DIVP: DIRECT. 0 = post divider (divide by N+1). “DIRECT”
15:8	RW	0x1	PLLC2_DIVN: PLL feedback divider. ” LATCHED”
1:0	RW	0x1	PLLC2_DIVM: PLL input divider. DIRECT. 00 = Divide by 1. 01 = Divide by 1. 10 = Divide by 2. 11 = Divide by 4.

5.7.190 CLK_RST_CONTROLLER_PLLC2_MISC_0_0

Offset: 0x4ec | Read/Write: R/W | Reset: 0x40000000 (0b01000000000000000000000000000000)

Bit	Reset	Description
31	0x0	PLLC2_STROBE: STROBE for various PLL input controls. Those controls are transparent only when STROBE is high.
30	0x1	PLLC2_RESET: Reset for digital logic of the PLL. DIRECT. 0 = Out of reset, 1 = Reset asserted. 0 = DISABLE 1 = ENABLE
29:28	0x0	PLLC2_SDM_DIV: SDM Clock Divide Ratio. LATCHED. 00 = Divide by 4. 01 = Divide by 8. 10 = Divide by 16. 11 = Divide by 32.
27:26	0x0	PLLC2_FILT_DIV: Filter Clock Divide Ratio. LATCHED. 00 = Divide by 8. 01 = Divide by 16. 10 = Divide by 32. 11 = Divide by 64.
25:18	0x0	PLLC2_ALPHA: IIR pole location. LATCHED.
17:9	0x0	PLLC2_KB: IIR filter feed-forward gain. LATCHED.
8:2	0x0	PLLC2_KA: IIR filter input gain. LATCHED.
1:0	0x0	PLLC2_KVCO: VCO gain adjustment. DIRECT.

5.7.191 CLK_RST_CONTROLLER_PLLC2_MISC_1_0

Offset: 0x4f0 | Read/Write: R/W | Reset: 0x08000000 (0bxxxx1000xx000000xx00xx00xxxx0000)

Bit	Reset	Description
27	0x1	PLLC2_IDDQ: 0 = OFF 1 = ON
26:24	0x0	PLLC2_VCO_BAND_SW: If calibration is not being done, this field can force the VCO band (select n+1 band, 1 to 8). LATCHED.
21:16	0x0	PLLC2_DAC_CODE_SW: For calibration at Vmin, the voltage is controlled by this field. LATCHED.
13:12	0x0	PLLC2_CAL_NUM_FRM: Number of frames during calibration for which the delta is collected for taking an average. LATCHED. 00 = 2, 01 = 4, 10 = 8, 11 = 16.
9	0x0	PLLC2_CAL_EN: Calibration is enabled (1) or disabled (0). DIRECT.
8	0x0	PLLC2_PD_AFTER_FD: Hold Phase-Detector output at '0' until Frequency Detector is locked. LATCHED. 1: Enable, 0: Disable
3:0	0x0	PLLC2_SDM_GAIN: SDM Gain Control. LATCHED.

5.7.192 CLK_RST_CONTROLLER_PLLC2_MISC_2_0

Offset: 0x4f4 | Read/Write: R/W | Reset: 0x00000000 (0bx000xx00xx00xx00xx00xx00xx00xx00)

Bit	Reset	Description
30	0x0	PLLC2_PTS: Base PLLC2 test output select.
29:28	0x0	PLLC2_PD_ULCK_FRM: pd_unlock_num_frames. Refer to the *NVIDIA Tegra K1 Mobile Processor Data Sheet* (DS-06742-001) for details. LATCHED.
25:24	0x0	PLLC2_PD_LCK_FRM: pd_lock_num_frames. Refer to the Tegra K1 data sheet for details. LATCHED.
21:20	0x0	PLLC2_PD_OUT_HYST: pd_pullout_hyst_val. Refer to the Tegra K1 data sheet for details. LATCHED.

Bit	Reset	Description
17:16	0x0	PLLC2_PD_IN_HYST: pd_pullin_hyst_val. Refer to the Tegra K1 data sheet for details. LATCHED.
13:12	0x0	PLLC2_FD_ULCK_FRM: fd_unlock_num_frames. Refer to the Tegra K1 data sheet for details. LATCHED
9:8	0x0	PLLC2_FD_LCK_FRM: fd_lock_num_frames. Refer to the Tegra K1 data sheet for details. LATCHED
5:4	0x0	PLLC2_FD_OUT_HYST: fd_pullout_hyst_val. Refer to the Tegra K1 data sheet for details. LATCHED.
1:0	0x0	PLLC2_FD_IN_HYST: fd_pullin_hyst_val. Refer to the Tegra K1 data sheet for details. LATCHED.

5.7.193 CLK_RST_CONTROLLER_PLLC2_MISC_3_0

Offset: 0x4f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	PLLC2_SETUP: Read the PLL data sheet for the description of each bit. DIRECT.

5.7.194 CLK_RST_CONTROLLER_PLLC3_BASE_0

Offset: 0x4fc | Read/Write: R/W | Reset: 0x0X000101 (0b000xxxxxx000xxxx00000001xxxxxx01)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLC3_BYPASS: DIRECT. 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLC3_ENABLE: DIRECT. 0 = disable, 1 = enable. Will invert and connect to IDDQ. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLC3_REF_DIS: 0 = enable reference clock, 1 = disable reference clock. To enter the minimum power mode for the PLL, this bit must be asserted. 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLC3_FREQ_LOCK: 0 = not locked, 1 = frequency is locked.
26	RO	X	PLLC3_PHASE_LOCK: 0 = not locked, 1 = phase is locked.
22:20	RW	0x0	PLLC3_DIVP: 0 = post divider (divide by N+1). DIRECT.
15:8	RW	0x1	PLLC3_DIVN: PLL feedback divider. LATCHED.
1:0	RW	0x1	PLLC3_DIVM: PLL input divider. DIRECT. 00 = Divide by 1, 01 = Divide by 1. 10 = Divide by 2. 11 = Divide by 4.

5.7.195 CLK_RST_CONTROLLER_PLLC3_MISC_0_0

Offset: 0x500 | Read/Write: R/W | Reset: 0x40000000 (0b00000000000000000000000000000000) | Default: 0x00644650

Bit	Reset	SW Default	Description
31	0x0	NONE	PLLC3_STROBE: STROBE for various PLL input controls. Those controls are transparent only when STROBE is high.
30	0x1	NONE	PLLC3_RESET: Reset for digital logic of the PLL. DIRECT. 0 = Out of reset, 1 = Reset asserted. 0 = DISABLE 1 = ENABLE

Bit	Reset	SW Default	Description
29:28	0x0	NONE	PLLC3_SDM_DIV: SDM Clock Divide Ratio. LATCHED. 00 = Divide by 4. 01 = Divide by 8. 10 = Divide by 16. 11 = Divide by 32.
27:26	0x0	NONE	PLLC3_FILT_DIV: Filter Clock Divide Ratio. LATCHED. 00 = Divide by 8. 01 = Divide by 16. 10 = Divide by 32. 11 = Divide by 64.
25:18	0x0	0x19	PLLC3_ALPHA: IIR pole location. LATCHED.
17:9	0x0	0x23	PLLC3_KB: IIR filter feed-forward gain. LATCHED.
8:2	0x0	0x14	PLLC3_KA: IIR filter input gain. LATCHED.
1:0	0x0	NONE	PLLC3_KVCO: VCO gain adjustment. DIRECT.

5.7.196 CLK_RST_CONTROLLER_PLLC3_MISC_1_0

Offset: 0x504 | Read/Write: R/W | Reset: 0x08000000 (0bxxxx1000xx000000xx00xx00xxxx0000)

Bit	Reset	Description
27	0x1	PLLC3_IDDQ: 0 = OFF 1 = ON
26:24	0x0	PLLC3_VCO_BAND_SW: If calibration is not being done, this field can force the VCO band (select n+1 band, 1 to 8). LATCHED.
21:16	0x0	PLLC3_DAC_CODE_SW: For calibration at Vmin, the voltage is controlled by this field. LATCHED.
13:12	0x0	PLLC3_CAL_NUM_FRM: Number of frames during calibration for which the delta is collected for taking an average. LATCHED. 00 = 2, 01 = 4, 10 = 8, 11 = 16.
9	0x0	PLLC3_CAL_EN: Calibration is enabled (1) or disabled (0). DIRECT.
8	0x0	PLLC3_PD_AFTER_FD: Hold Phase-Detector output at '0' until Frequency Detector is locked. 1: Enable, 0: Disable. LATCHED.
3:0	0x0	PLLC3_SDM_GAIN: SDM Gain Control. LATCHED.

5.7.197 CLK_RST_CONTROLLER_PLLC3_MISC_2_0

Offset: 0x508 | Read/Write: R/W | Reset: 0x00000000 (0bx000xx00xx00xx00xx00xx00xx00xx00)

Bit	Reset	Description
30	0x0	PLLC3_PTS: Base PLLC3 test output select.
29:28	0x0	PLLC3_PD_ULCK_FRM: pd_unlock_num_frames. Refer to the *NVIDIA Tegra K1 Mobile Processor Data Sheet* (DS-06742-001) for details. LATCHED.
25:24	0x0	PLLC3_PD_LCK_FRM: pd_lock_num_frames. Refer to the Tegra K1 data sheet for details. LATCHED.
21:20	0x0	PLLC3_PD_OUT_HYST: pd_pullout_hyst_val. Refer to the Tegra K1 data sheet for details. LATCHED.
17:16	0x0	PLLC3_PD_IN_HYST: pd_pullin_hyst_val. Refer to the Tegra K1 data sheet for details. LATCHED.
13:12	0x0	PLLC3_FD_ULCK_FRM: fd_unlock_num_frames. Refer to the Tegra K1 data sheet for details. LATCHED.
9:8	0x0	PLLC3_FD_LCK_FRM: fd_lock_num_frames. Refer to the Tegra K1 data sheet for details. LATCHED.
5:4	0x0	PLLC3_FD_OUT_HYST: fd_pullout_hyst_val. Refer to the Tegra K1 data sheet for

Bit	Reset	Description
		details. LATCHED.
1:0	0x0	PLLC3_FD_IN_HYST: fd_pullin_hyst_val. Refer to the Tegra K1 data sheet for details. LATCHED.

5.7.198 CLK_RST_CONTROLLER_PLLC3_MISC_3_0

Offset: 0x50c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	PLLC3_SETUP: Refer to the PLL datasheet for the descriptions of each bit. DIRECT.

5.7.199 CLK_RST_CONTROLLER_PLLX_MISC_1_0

Offset: 0x510 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000000)

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

5.7.200 CLK_RST_CONTROLLER_PLLX_MISC_2_0

Offset: 0x514 │ Read/Write: R/W │ Reset: 0x0000000X (0b00000000000000000000000xx00xx00)

Bit	R/W	Reset	Description
31:24	RW	0x0	PLLX_DYNRAMP_STEPB: MISC_2 PLLX DYNRAMP
23:16	RW	0x0	PLLX_DYNRAMP_STEPA: MISC_2 PLLX DYNRAMP
15:8	RW	0x0	PLLX_NDIV_NEW: MISC_2 PLLX DYNRAMP
5	RW	0x0	PLLX_EN_FSTLCK: MISC_2 PLLX DYNRAMP
4	RW	0x0	PLLX_LOCK_OVERRIDE: MISC_2 PLLX DYNRAMP
3	RO	X	PLLX_PLL_FREQLOCK: MISC_2 PLLX DYNRAMP
2	RO	X	PLLX_DYNRAMP_DONE: MISC_2 PLLX DYNRAMP
1	RW	0x0	PLLX_CLAMP_NDIV: MISC_2 PLLX DYNRAMP
0	RW	0x0	PLLX_EN_DYNRAMP: MISC_2 PLLX DYNRAMP

5.7.201 CLK_RST_CONTROLLER_PLLX_MISC_3_0

Offset: 0x518 │ Read/Write: R/W │ Reset: 0x00000008 (0bxxxxxxxx00000000xxxx0000xxxx1000)

Bit	Reset	Description
23:16	0x0	PLLX_PRB_OBS_SEL: 0 = CSITE 1 = FCPU0_LEAF 2 = FCPU1_LEAF 3 = FCPU2_LEAF 4 = FCPU3_LEAF 5 = MC 6 = MSELECT 7 = FTOP_SHAPER
11:10	0x0	PLLX_PRB_DRV_CTRL
9:8	0x0	PLLX_SEL_PRB
3	0x1	PLLX_IDDQ:0:

Bit	Reset	Description
		The PLLX is powered up. 1:Software can put the PLLX in IDDQ by setting this bit.
2:1	0x0	PLLX_KCP: Charge Pump Gain control
0	0x0	PLLX_KVCO:VCO gain

5.7.202 CLK_RST_CONTROLLER_XUSBIO_PLL_CFG0_0

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

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

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

Offset: 0x51c │ Read/Write: R/W │ Reset: 0x0X00000d (0bxxxxxx10xxxxxxxxxxxxxxxxx0001101)

Bit	R/W	Reset	Description
27:26	RO	X	XUSBIO_SEQ_STATE: XUSBIO power sequencer state 0 = SEQ_OFF 1 = SEQ_ON 2 = SEQ_BUSY
25	RW	SEQ_ON	XUSBIO_SEQ_START_STATE: XUSBIO power sequencer start state 0 = SEQ_OFF 1 = SEQ_ON
24	RW	DISABLE	XUSBIO_SEQ_ENABLE: XUSBIO power sequencer enable. Start state is loaded on first cycle after set. 0 = DISABLE 1 = ENABLE
6	RW	0x0	XUSBIO_PADPLL_USE_LOCKDET: 0=Use programmable delay, 1=use lockdet signals from PLL
5	RW	0x0	XUSBIO_SEQ_RESET_INPUT_VALUE: 0=XUSBIO PLL ON, 1=XUSB IOPLL OFF. Software controls the state machine by setting this and the XUSBIO_SEQ_IN_SWCTL bit
4	RW	0x0	XUSBIO_SEQ_IN_SWCTL: 0=seq input by hardware, 1=seq input by software.
3	RW	ENABLE	XUSBIO_CLK_ENABLE_OVERRIDE_VALUE: Override value used only when XUSBIO_CLK_ENABLE_SWCTL is set 0 = DISABLE 1 = ENABLE
2	RW	0x1	XUSBIO_CLK_ENABLE_SWCTL: 0=ioclks enabled by hardware, 1=ioclks enabled by software
1	RW	0x0	XUSBIO_PADPLL_RESET_OVERRIDE_VALUE: 0=XUSBIO PLL on, 1=PLL reset. Override value used only when XUSBIO_PADPLL_RESET_SWCTL is set
0	RW	0x1	XUSBIO_PADPLL_RESET_SWCTL: 0=Reset by hardware, 1=Reset by software

5.7.203 CLK_RST_CONTROLLER_XUSBIO_PLL_CFG1_0

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

Offset: 0x520 │ Read/Write: R/W │ Reset: 0x000a0a0f (0bxxxxxxxx00001010000010100000 1111)

Bit	Reset	Description
23:16	0xa	XUSBIO_PADPLL_RESET2_PADPLL_IDDQ_DLY: PT1: Delay from XUSBIO PAD PLL RESET assertion to the PAD PLL IDDQ assertion. Range is 0 - 10 µs.

Bit	Reset	Description
15:8	0xa	XUSBIO_PADPLL_IDDQ2_PADPLL_RESET_DLY: PT2: Delay from XUSBIO PAD PLL IDDQ de-assertion to the PAD PLL reset de-assertion. Range is 0 - 10 µs.
7:0	0xf	XUSBIO_PADPLL_PU_POST_DLY: PT3: Delay from XUSBIO PAD PLL Reset de-assertion to the PLL LOCK. Range is 0 -15 µs.

5.7.204 CLK_RST_CONTROLLER_PLLE_AUX1_0

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

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

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

5.7.205 CLK_RST_CONTROLLER_PLLP_RESHIFT_0

Offset: 0x528 │ Read/Write: R/W │ Reset: 0x0000003b (0bxxxxxxxxxxxxxxxxxxxxxxxx0000111011)

Bit	Reset	Description
9:2	0xe	PLLP_OUT0_RATIO: PLLP_OUT0 divider from base PLLP (lsb denotes 0.5x). The default is 408/8=51 MHz.
1	ENABLE	PLLP_OUT0_CLKEN: PLLP_OUT0 divider clock enable. 0 = DISABLE 1 = ENABLE
0	RESET_DISABLE	PLLP_OUT0_RSTN: PLLP_OUT0 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

5.7.206 CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0

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

Bit	R/W	Reset	Description
31	RO	X	UTMIPLL_LOCK: 0 = not lock, 1 = lock. (Phase and Frequency)
27:26	RO	X	UTMIPLL_SEQ_STATE: UTMIPLL power sequencer state 0 = SEQ_OFF 1 = SEQ_ON 2 = SEQ_BUSY
25	RW	SEQ_ON	UTMIPLL_SEQ_START_STATE: UTMIPLL power sequencer start state 0 = SEQ_OFF 1 = SEQ_ON
24	RW	DISABLE	UTMIPLL_SEQ_ENABLE: UTMIPLL power sequencer enable. Start state is loaded on first cycle after set. 0 = DISABLE 1 = ENABLE
7	RW	0x0	UTMIPLL_IDDQ_PD_INCLUDE: While powering down through the HW state machine, puts the PLL in IDDQ as well to save more power.
6	RW	0x0	UTMIPLL_USE_LOCKDET: 0=Use programmable delay, 1=use lockdet signals from PLL and programmable delay on top of that.

Bit	R/W	Reset	Description
5	RW	0x0	UTMIPLL_SEQ_RESET_INPUT_VALUE: 0=UTMIPLL ON, 1=UTMIPLL OFF. Software controls the state machine by setting this and UTMIPLL_SEQ_IN_SWCTL bit
4	RW	0x0	UTMIPLL_SEQ_IN_SWCTL: 0=seq input by hardware, 1=seq input by software
3	RW	ENABLE	UTMIPLL_CLK_ENABLE_OVERRIDE_VALUE: Override value used only when UTMIPLL_CLK_ENABLE_SWCTL is set 0 = DISABLE 1 = ENABLE
2	RW	0x1	UTMIPLL_CLK_ENABLE_SWCTL: 0=UTMIP clocks enable by hardware, 1=UTMIP clocks enable by software
1	RW	0x1	UTMIPLL_IDDQ_OVERRIDE_VALUE: 0=PLL not in IDDQ mode, 1=PLL in IDDQ mode. Override value used only when UTMIPLL_IDDQ_SWCTL is set
0	RW	0x1	UTMIPLL_IDDQ_SWCTL: 0=IDDQ by hardware, 1=IDDQ by software.

5.7.207 CLK_RST_CONTROLLER_PLLU_HW_PWRDN_CFG0_0

Offset: 0x530 │ Read/Write: R/W │ Reset: 0x0X00008d (0bxxxxxx10xxxxxxxxxxxxxxxx100011x1)

Bit	R/W	Reset	Description
27:26	RO	X	PLLU_SEQ_STATE: PLLU power sequencer state 0 = SEQ_OFF 1 = SEQ_ON 2 = SEQ_BUSY
25	RW	SEQ_ON	PLLU_SEQ_START_STATE: PLLU power sequencer start state 0 = SEQ_OFF 1 = SEQ_ON
24	RW	DISABLE	PLLU_SEQ_ENABLE: PLLU power sequencer enable. Start state is loaded on the first cycle after set. 0 = DISABLE 1 = ENABLE
7	RW	0x1	PLLU_USE_SWITCH_DETECT: 0=Use programmable delay, 1=Use hardware switch detect logic.
6	RW	0x0	PLLU_USE_LOCKDET: 0=Use programmable delay, 1=Use lockdet signals from PLL and programmable delay on top of that.
5	RW	0x0	PLLU_SEQ_RESET_INPUT_VALUE: 0=PLLU ON, 1=PLLU OFF. Software controls the state machine by setting this and the PLLU_SEQ_IN_SWCTL bit
4	RW	0x0	PLLU_SEQ_IN_SWCTL: 0=Seq input by hardware, 1=Seq input by software
3	RW	ENABLE	PLLU_CLK_ENABLE_OVERRIDE_VALUE: Override value used only when PLLU_CLK_ENABLE_SWCTL is set 0 = DISABLE 1 = ENABLE
2	RW	0x1	PLLU_CLK_ENABLE_SWCTL: 0=PLLU clocks enabled by hardware, 1=PLLU clocks enabled by software
0	RW	0x1	PLLU_CLK_SWITCH_SWCTL: 0=Control SS/FS clock frequency through hardware, 1=Control FS/SS clock frequency through software

5.7.208 CLK_RST_CONTROLLER_XUSB_PLL_CFG0_0

Note that default value is the recommended range.

Offset: 0x534 | Read/Write: R/W | Reset: 0x80191480 (0b10000000000110010001010010000000)

Bit	Reset	Description
31:24	0x80	PLLU_CLK_SWITCH_DLY: Delay from SS Clock source change to the actual frequency change to 32 kHz clock. Range is ~100 µs.
23:14	0x64	PLLU_LOCK_DLY: Delay from PLLU ENABLE assertion to the PLL LOCK. Range is 100 µs-1 ms.
13:10	0x5	UTMIPLL_IDDQ2_ENABLE_DLY: Delay from UTMIPLL IDDQ de-assertion to the UTMIPLL ENABLE assertion. Range ~5 µs.
9:0	0x80	UTMIPLL_LOCK_DLY: Delay from UTMIPLL ENABLE assertion to the PLL LOCK. Range 5-15 µs.

5.7.209 CLK_RST_CONTROLLER_CLK_CPU_MISC_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

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

5.7.210 CLK_RST_CONTROLLER_CLK_CPUG_MISC_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

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

5.7.211 CLK_RST_CONTROLLER_CLK_CPULP_MISC_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

Offset: 0x544 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx010)

Bit	Reset	Description
2:0	0x2	CPULP_ATCLK_RATIO: Clock divider ratio for the ATB clocks in CCPLEX 000 = div-by-2. 001 = div-by-3. 010 = div-by-4.

Bit	Reset	Description
		011 = div-by-5. 100 = div-by-6. Other values = Reserved

5.7.212 CLK_RST_CONTROLLER_PLLX_HW_CTRL_CFG_0

Offset: 0x548 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000001)

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

5.7.213 CLK_RST_CONTROLLER_PLLX_SW_RAMP_CFG_0

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

Bit	Reset	Description
31:24	0x0	DYNRAMP_STEPA: PLL ramp rate
23:16	0x0	DYNRAMP_STEPB: PLL ramp rate
15:8	0x0	NDIV_NEW: PLL ramp rate
7:0	0x0	NDIV: PLL ramp rate

5.7.214 CLK_RST_CONTROLLER_PLLX_HW_CTRL_STATUS_0

Offset: 0x550 | Read/Write: RO | Reset: 0xXX000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	FSM_STATE: 0 = DISABLE 1 = SEQ_AWAIT 2 = SEQ_FAST_BEGIN 3 = SEQ_FAST_BUSY0 4 = SEQ_FAST_BUSY1 5 = SEQ_FAST_BUSY2 6 = SEQ_FAST_BUSY3 7 = SEQ_SLOW_BEGIN 8 = SEQ_SLOW_BUSY0

Bit	Reset	Description
		9 = SEQ_SLOW_BUSY1 10 = SEQ_SLOW_BUSY2 11 = SEQ_SLOW_BUSY3 12 = SEQ_DONE 13 = ILLEGAL
27:26	X	CFG_REG_SELECT: 0 = NONE 1 = HW 2 = SW
10	X	HW_RAMP_DONE: 0 = FALSE 1 = TRUE
9	X	SW_RAMP_DONE: 0 = FALSE 1 = TRUE
8	X	LONG_LATENCY_THROTTLE: 0 = DISABLE 1 = ENABLE
7	X	HW_RESTORE_EN: 0 = DISABLE 1 = ENABLE
6	X	HW_THROTTLE_EN: 0 = DISABLE 1 = ENABLE
5	X	SW_RAMP_STATUS: 0 = DONE 1 = INPROGRESS
4	X	HW_RAMP_STATUS: 0 = DONE 1 = INPROGRESS
3:2	X	SEQ_STATUS: 0 = SEQ_DIABLE 1 = SEQ_ENABLE 2 = SEQ_BUSY_FAST_MODE 3 = SEQ_BUSY_SLOW_MODE
1	X	SW_OVERRIDE_STATUS: 0 = DISABLE 1 = ENABLE
0	X	SWCTL_STATUS: 0 = DISABLE 1 = ENABLE

5.7.215 CLK_RST_CONTROLLER_SPARE_REG0_0

Offset: 0x55c │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:6	0x0	VAL
5	0x0	DIVIDER_FSM_CYA: Disable ability to change rate and ratio while module clock is off. Module should be off when changing this value.
4	0x0	VAL1
3:2	0x0	CLK_M_DIVISOR: N = Divide by (n+1).
1	0x0	TMR_CLKEN_STICKY
0	0x0	EMC_LATENCY_OVERRIDE

5.7.216 CLK_RST_CONTROLLER_PLLD2_SS_CFG_0

Offset: 0x570 | Read/Write: R/W | Reset: 0x1XX00000 (0b00010xxxx0xxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLD2_EN_SDM:Fractional N divider enable 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLD2_EN_SSC: Spread spectrum enable 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
29	RW	0x0	PLLD2_EN_DITHER2
28	RW	0x1	PLLD2_EN_DITHER
27	RW	0x0	PLLD2_SDM_RESET
25:23	RO	X	PLLD2_SDM_TEST_OUT
22	RW	0x0	PLLD2_CLAMP

5.7.217 CLK_RST_CONTROLLER_PLLD2_SS_CTRL1_0

Offset: 0x574 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PLLD2_SDM_SSC_MAX
15:0	0x0	PLLD2_SDM_SSC_MIN

5.7.218 CLK_RST_CONTROLLER_PLLD2_SS_CTRL2_0

Offset: 0x578 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PLLD2_SDM_SSC_STEP
15:0	0x0	PLLD2_SDM_DIN

5.7.219 CLK_RST_CONTROLLER_PLLDP_BASE_0

PLLDP Registers

Table 22: PLDIV - Divider Control for CLKOUT

PLDIV[3:0]	CLKOUT
0000	vcoclock/1 (pdivider is powered down)
0001	vcoclock/2
0010	vcoclock/3
0011	vcoclock/4

PLDIV[3:0]	CLKOUT
0100	vcoclock/5
0101	vcoclock/6
0110	vcoclock/8
0111	vcoclock/10
1000	vcoclock/12
1001	vcoclock/16
1010	vcoclock/12
1011	vcoclock/16
1100	vcoclock/20
1101	vcoclock/24
1110	vcoclock/32
1111	Reserved

vcoclock Frequency:

- Normal mode: EN_SDM=DISABLE, SDM_RESET=0

  Fvco = Fref / MDIV * NDIV
- Fractional NDIV mode: EN_SDM=ENABLE, EN_SSC=DISABLE, SDM_RESET=0

  Fvco = Fref / MDIV * (NDIV + 0.5 + SDM_DIN / 8192)
- Spread Spectrum mode: EN_SDM=ENABLE, EN_SSC=ENABLE, SDM_RESET=0

  Fvco_max = Fref / MDIV * (NDIV + 0.5 + SDM_SSC_MAX / 8192), SDM_SSC_MAX < 0x300

  Fvco_min = Fref / MDIV * (NDIV + 0.5 + SDM_SSC_MIN / 8192), SDM_SSC_MIN > -0x300 (0xD000)

  Fvco_step = 2 * (Fmod * MDIV / Fref) * (SDM_SSC_MAX-SDM_SSC_MIN), Fmod-modulation frequency,e.g., 33KHz

  = SDM_SSC_STEP[11:0] / (SDM_SSC_STEP[15:12] + 1)

Offset: 0x590 | Read/Write: R/W | Reset: 0xXX080000 (0b000xx00000001xx00000000000000000)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLDP_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLDP_ENABLE: 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
29	RW	REF_ ENABLE	PLLDP_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 0 = REF_ENABLE 1 = REF_DISABLE
28	RO	X	PLLDP_FREQLOCK: 0 = not lock, 1 = lock freq

Bit	R/W	Reset	Description
27	RO	X	PLLDP_LOCK: 0 = not lock, 1 = lock freq + phase
26:25	RW	0x0	PLLDP_REF_SRC_SEL:Reference source select sel[0]=1 => ref_src = PLLREFE_CLKOUTsel[0]=0 && sel[1]=1 => ref_src = pllP_out0sel[0]=0 && sel[1]=0 => ref_src = osc_div_clk
24	RW	0x0	PLLDP_LOCK_OVERRIDE:Forces PLL_LOCK to 1
23:20	RW	0x0	PLLDP_PLDIV: PL divider
19	RW	0x1	PLLDP_IDDQ: 0 = The PLL is powered up, 1 = Software can put the PLL in IDDQ by setting this bit 0 = OFF 1 = ON
16	RW	0x0	PLLDP_PTS: Base PLLDP test output select.
15:8	RW	0x0	PLLDP_NDIV:N divider
7:0	RW	0x0	PLLDP_MDIV:M divider

5.7.220 CLK_RST_CONTROLLER_PLLDP_MISC_0

Offset: 0x594 | Read/Write: R/W | Reset: 0xXX000000 (0b00xxx00000000000000000000000000)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLDP_EN_FSTLCK:Enables Fast lock 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLDP_LOCK_ENABLE: 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
29:27	RO	X	PLLDP_MON_TEST_OUT
26:25	RW	0x0	PLLDP_KCP: Charge pump gain control
24	RW	0x0	PLLDP_KVCO:VCO gain
23:0	RW	0x0	PLLDP_SETUP:setup[23:0]

5.7.221 CLK_RST_CONTROLLER_PLLDP_SS_CFG_0

Offset: 0x598 | Read/Write: R/W | Reset: 0x1XX00000 (0b00010xxxx0xxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLDP_EN_SDM: Fractional N divider enable 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLDP_EN_SSC: Spread spectrum enable 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
29	RW	0x0	PLLDP_EN_DITHER2
28	RW	0x1	PLLDP_EN_DITHER
27	RW	0x0	PLLDP_SDM_RESET
25:23	RO	X	PLLDP_SDM_TEST_OUT
22	RW	0x0	PLLDP_CLAMP

5.7.222 CLK_RST_CONTROLLER_PLLDP_SS_CTRL1_0

Offset: 0x59c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PLLDP_SDM_SSC_MAX
15:0	0x0	PLLDP_SDM_SSC_MIN

5.7.223 CLK_RST_CONTROLLER_PLLDP_SS_CTRL2_0

Offset: 0x5a0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PLLDP_SDM_SSC_STEP
15:0	0x0	PLLDP_SDM_DIN

5.7.224 CLK_RST_CONTROLLER_PLLC4_BASE_0

Offset: 0x5a4 | Read/Write: R/W | Reset: 0xXX080101 (0b000xx00000001xx00000000100000001)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLC4_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLC4_ENABLE: 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLC4_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 0 = REF_ENABLE 1 = REF_DISABLE
28	RO	X	PLLC4_FREQLOCK: 0 = not lock, 1 = lock feq
27	RO	X	PLLC4_LOCK: 0 = not lock, 1 = lock feq + phase
26:25	RW	0x0	PLLC4_REF_SRC_SEL:Reference source select sel[0]=1 => ref_src = PLLREFE_CLKOUTsel[0]=0 && sel[1]=1 => ref_src = pllP_out0sel[0]=0 && sel[1]=0 => ref_src = osc_div_clk
24	RW	0x0	PLLC4_LOCK_OVERRIDE:Forces PLL_LOCK to 1
23:20	RW	0x0	PLLC4_PLDIV:PL divider
19	RW	0x1	PLLC4_IDDQ: 0 The PLL is powered up, 1 Software can put the PLL in IDDQ by setting this bit 0 = OFF 1 = ON
16	RW	0x0	PLLC4_PTS: Base PLLC4 test output select.
15:8	RW	0x1	PLLC4_NDIV:N divider
7:0	RW	0x1	PLLC4_MDIV:M divider

5.7.225 CLK_RST_CONTROLLER_PLLC4_MISC_0

Offset: 0x5a8 | Read/Write: R/W | Reset: 0xXX000000 (0b00xxx00000000000000000000000000)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLC4_EN_FSTLCK:Enables Fast lock 0 = DISABLE

Bit	R/W	Reset	Description
			1 = ENABLE
30	RW	DISABLE	PLLC4_LOCK_ENABLE: 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
29:27	RO	X	PLLC4_MON_TEST_OUT
26:25	RW	0x0	PLLC4_KCP:Charge pump gain control
24	RW	0x0	PLLC4_KVCO:VCO gain
23:0	RW	0x0	PLLC4_SETUP:setup[23:0]

5.7.226 CLK_RST_CONTROLLER_PLLC4_SS_CFG_0

Offset: 0x5ac | Read/Write: R/W | Reset: 0x1XX00000 (0b00010xxxx0xxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLC4_EN_SDM: 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLC4_EN_SSC: 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
29	RW	0x0	PLLC4_EN_DITHER2
28	RW	0x1	PLLC4_EN_DITHER
27	RW	0x0	PLLC4_SDM_RESET
25:23	RO	X	PLLC4_SDM_TEST_OUT
22	RW	0x0	PLLC4_CLAMP

5.7.227 CLK_RST_CONTROLLER_PLLC4_SS_CTRL1_0

Offset: 0x5b0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PLLC4_SDM_SSC_MAX

Bit	Reset	Description
15:0	0x0	PLLC4_SDM_SSC_MIN

5.7.228 CLK_RST_CONTROLLER_PLLC4_SS_CTRL2_0

Offset: 0x5b4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PLLC4_SDM_SSC_STEP
15:0	0x0	PLLC4_SDM_DIN

5.7.229 CLK_RST_CONTROLLER_CLK_SPARE0_0

Offset: 0x5c4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RESERVED

5.7.230 CLK_RST_CONTROLLER_CLK_SPARE1_0

Offset: 0x5c8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RESERVED

5.7.231 CLK_RST_CONTROLLER_GPU_ISOB_CTRL_0

Offset: 0x5cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	DISABLE	CAR_ISOB_EN: Enable/Disable the idle slowdown on boot (ISoB) to the GPU 0 = DISABLE 1 = ENABLE

5.7.232 CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_CORE_HOST_0

Offset: 0x600 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	XUSB_CORE_HOST_CLK_SRC: 000 = clk_m, 001 = pllP_out0, 010 = pllC2_out0, 011 = pllC_out0, 100 = pllC3_out0, 101 = pllRefe_OUT 0 = CLK_M 1 = PLLP_OUT0 2 = PLLC2_OUT0 3 = PLLC_OUT0 4 = PLLC3_OUT0 5 = PLLREFE_CLKOUT
7:0	0x0	XUSB_CORE_HOST_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.233 CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_FALCON_0

Offset: 0x604 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	XUSB_FALCON_CLK_SRC: 000 = clk_m, 001 = pllP_out0, 010 = pllC2_out0, 011 = pllC_out0, 100 = pllC3_out0, 101 = pllRefe_OUT 0 = CLK_M 1 = PLLP_OUT0 2 = PLLC2_OUT0 3 = PLLC_OUT0 4 = PLLC3_OUT0 5 = PLLREFE_CLKOUT
7:0	0x0	XUSB_FALCON_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.234 CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_FS_0

Offset: 0x608 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	XUSB_FS_CLK_SRC: 000 = clk_m, 010 = FO_48M, 100 = pllP_out0, 110 = HSIC_480 0 = CLK_M 2 = FO_48M 4 = PLLP_OUT0 6 = HSIC_480
7:0	0x0	XUSB_FS_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.235 CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_CORE_DEV_0

Offset: 0x60c | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	XUSB_CORE_DEV_CLK_SRC: 000 = clk_m, 001 = pllP_out0, 010 = pllC2_out0, 011 = pllC_out0, 100 = pllC3_out0, 101 = pllRefe_OUT 0 = CLK_M 1 = PLLP_OUT0 2 = PLLC2_OUT0 3 = PLLC_OUT0 4 = PLLC3_OUT0 5 = PLLREFE_CLKOUT
7:0	0x0	XUSB_CORE_DEV_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.236 CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_SS_0

Offset: 0x610 | Read/Write: R/W | Reset: 0x00000000 (0b000xxx00xxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	XUSB_SS_CLK_SRC: 000 = clk_m, 001 = pllRefe_OUT, 010 = clk_s, 011 = HSIC_480, 100 = pllC_out0, 101 = pllC2_out0, 110 = pllC3_out0, 111 = osc_div 0 = CLK_M 1 = PLLREFE_CLKOUT 2 = CLK_S 3 = HSIC_480 4 = PLLC_OUT0 5 = PLLC2_OUT0 6 = PLLC3_OUT0 7 = OSC_DIV
25	0x0	XUSB_HS_CLK_BYPASS_SWITCH: 0 = HS clock is switch/dividers output 1 = HS Clock is derived from PLLU 60M output directly.
24	0x0	XUSB_SS_CLK_BYPASS_SWITCH: 0 = SS clock is switch/dividers output 1 = SS Clock is derived from osc_div clock directly.
7:0	0x0	XUSB_SS_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.237 CLK_RST_CONTROLLER_CLK_SOURCE_CILAB_0

Offset: 0x614 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	CILAB_CLK_SRC: 000 = pllP_out0, 010 = pllC_out0, 110 = clk_m 0 = PLLP_OUT0 2 = PLLC_OUT0 6 = CLK_M
7:0	0x0	CILAB_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.238 CLK_RST_CONTROLLER_CLK_SOURCE_CILCD_0

Offset: 0x618 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	CILCD_CLK_SRC: 000 = pllP_out0, 010 = pllC_out0, 110 = clk_m 0 = PLLP_OUT0 2 = PLLC_OUT0 6 = CLK_M
7:0	0x0	CILCD_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.239 CLK_RST_CONTROLLER_CLK_SOURCE_CILE_0

Offset: 0x61c | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	CILE_CLK_SRC: 000 = pllP_out0, 010 = pllC_out0, 110 = clk_m 0 = PLLP_OUT0 2 = PLLC_OUT0 6 = CLK_M
7:0	0x0	CILE_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.240 CLK_RST_CONTROLLER_CLK_SOURCE_DSIA_LP_0

Offset: 0x620 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	DSIA_LP_CLK_SRC: 000 = pllP_out0, 010 = pllC_out0, 110 = clk_m 0 = PLLP_OUT0 2 = PLLC_OUT0 6 = CLK_M
7:0	0x0	DSIA_LP_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.241 CLK_RST_CONTROLLER_CLK_SOURCE_DSIB_LP_0

Offset: 0x624 | Read/Write: R/W | Reset: 0xc0000000 (0b11xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	DSIB_LP_CLK_SRC: 000 = pllP_out0, 010 = pllC_out0, 110 = clk_m 0 = PLLP_OUT0 2 = PLLC_OUT0 6 = CLK_M
7:0	0x0	DSIB_LP_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.242 CLK_RST_CONTROLLER_CLK_SOURCE_ENTROPY_0

Offset: 0x628 | Read/Write: R/W | Reset: 0x20000000 (0b001xxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
31:29	CLK_M	ENTROPY_CLK_SRC: 000 = pllP_out0, 001 = clk_m, 010 = clk_s, 011 = pllE_out0 0 = PLLP_OUT0 1 = CLK_M 2 = CLK_S 3 = PLLE_OUT0
8	0x0	ENTROPY_CLK_LOCK: 1 = Lock entire CLK_SOURCE_ENTROPY register, can only be unlocked by reset.
7:0	0x0	ENTROPY_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.243 CLK_RST_CONTROLLER_CLK_SOURCE_DVFS_REF_0

Offset: 0x62c | Read/Write: R/W | Reset: 0xd0000000 (0b1101xxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	DVFS_REF_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
28	0x1	DVFS_REF_MASTER_CLKEN: Reserved.
7:0	0x0	DVFS_REF_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.244 CLK_RST_CONTROLLER_CLK_SOURCE_DVFS_SOC_0

Offset: 0x630 | Read/Write: R/W | Reset: 0xd0000000 (0b1101xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	DVFS_SOC_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
28	0x1	DVFS_SOC_MASTER_CLKEN: Reserved.
7:0	0x0	DVFS_SOC_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.245 CLK_RST_CONTROLLER_CLK_SOURCE_TRACECLKIN_0

Offset: 0x634 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	TRACECLKIN_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	TRACECLKIN_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.246 CLK_RST_CONTROLLER_CLK_SOURCE_ADX0_0

Offset: 0x638 | Read/Write: R/W | Reset: 0xd0000000 (0b1101xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	ADX0_CLK_SRC: 000 = pllA_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = clk_m 0 = PLLA_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = CLK_M
28	0x1	ADX0_MASTER_CLKEN: Reserved.
7:0	0x0	ADX0_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.247 CLK_RST_CONTROLLER_CLK_SOURCE_AMX0_0

Offset: 0x63c | Read/Write: R/W | Reset: 0xd0000000 (0b1101xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	AMX0_CLK_SRC: 000 = pllA_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = clk_m 0 = PLLA_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = CLK_M
28	0x1	AMX0_MASTER_CLKEN: Reserved.
7:0	0x0	AMX0_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.248 CLK_RST_CONTROLLER_CLK_SOURCE_EMC_LATENCY_0

Offset: 0x640 | Read/Write: R/W | Reset: 0x60000000 (0b011xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	EMC_LATENCY_CLK_SRC: 000 = pllM_out0, 001 = pllC_out0, 010 = pllP_out0, 011 = clk_m, 100 = pllM_out_for_emc, same as PLLM, 101 = pllC2_out, 110 = pllC3_out, 111 = empty 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = CLK_M 4 = PLLM_UD 5 = PLLC2_OUT0 6 = PLLC3_OUT0
7:0	0x0	EMC_LATENCY_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.249 CLK_RST_CONTROLLER_CLK_SOURCE_SOC_THERM_0

Offset: 0x644 | Read/Write: R/W | Reset: 0x40000000 (0b010xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLP_OUT0	SOC_THERM_CLK_SRC: 000 = pllM_out0, 001 = pllC_out0, 010 = pllP_out0, 011 = pllA_out0, 100 = pllC2_out0, 101 = pllC3_out0 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA_OUT0 4 = PLLC2_OUT0 5 = PLLC3_OUT0
7:0	0x0	SOC_THERM_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

5.7.250 CLK_RST_CONTROLLER_CLK_SOURCE_VI_SENSOR2_0

Offset: 0x658 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLM_OUT0	VI_SENSOR2_CLK_SRC: 000 = pllM_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 110 = pllA_out0 0 = PLLM_OUT0

Bit	Reset	Description
		1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = PLLA_OUT0
7:0	0x0	VI_SENSOR2_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.7.251 CLK_RST_CONTROLLER_CLK_SOURCE_I2C6_0

Offset: 0x65c | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31:29	CLK_M	I2C6_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
15:0	0x0	I2C6_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 1.0x)

5.7.252 CLK_RST_CONTROLLER_CLK_SOURCE_EMC_DLL_0

Offset: 0x664 | Read/Write: R/W | Reset: 0x60000000 (0b011xxxxxxxxxxx0xxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	EMC_DLL_CLK_SRC: 000 = pllM_out0, 001 = pllC_out0, 010 = pllP_out0, 011 = clk_m, 100 = pllM_out_for_emc, same as pllM_out0, 101 = pllC2_out, 110 = pllC3_out, 111 = empty 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = CLK_M 4 = PLLM_UD 5 = PLLC2_OUT0 6 = PLLC3_OUT0
16	0x0	EMC_DLL_DYN_MUX_CTRL: 0 = CDB is the source for DLL clock. 1 = The output of this switch is the source for DLL clock.
7:0	0x0	EMC_DLL_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.7.253 CLK_RST_CONTROLLER_CLK_SOURCE_HDMI_AUDIO_0

Offset: 0x668 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLP_OUT0	HDMI_AUDIO_CLK_SRC: 000 = pllP_out0, 001 = pllC_out0, 010 = pllC2_out0, 011 = CLK_M 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLC2_OUT0 3 = CLK_M
7:0	0x0	HDMI_AUDIO_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.7.254 CLK_RST_CONTROLLER_CLK_SOURCE_CLK72MHZ_0

Offset: 0x66c | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLP_OUT3	CLK72MHZ_CLK_SRC: 000 = pllP_out3 001 = pllC_out0, 010 = pllC2_out0, 011 = CLK_M 0 = PLLP_OUT3 1 = PLLC_OUT0 2 = PLLC2_OUT0 3 = CLK_M
7:0	0x0	CLK72MHZ_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.7.255 CLK_RST_CONTROLLER_CLK_SOURCE_ADX1_0

Offset: 0x670 | Read/Write: R/W | Reset: 0xd0000000 (0b1101xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	ADX1_CLK_SRC: 000 = pllA_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = clk_m 0 = PLLA_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = CLK_M
28	0x1	ADX1_MASTER_CLKEN: Obsolete. Use CLK_ENB_ADX1 to control clock enable
7:0	0x0	ADX1_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.7.256 CLK_RST_CONTROLLER_CLK_SOURCE_AMX1_0

Offset: 0x674 | Read/Write: R/W | Reset: 0xd0000000 (0b1101xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	AMX1_CLK_SRC: 000 = pllA_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = clk_m 0 = PLLA_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = CLK_M
28	0x1	AMX1_MASTER_CLKEN: Obsolete. Use CLK_ENB_AMX1 to control clock enable
7:0	0x0	AMX1_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.7.257 CLK_RST_CONTROLLER_CLK_SOURCE_VIC_0

Offset: 0x678 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31:29	PLLM_OUT0	VIC_CLK_SRC: 000 = pllM_out0, 001 = pllC_out0, 010 = pllP_out0, 011 = pllA_out0, 100 = pllC2_out0, 101 = pllC3_out0, 110 = clk_m 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA_OUT0 4 = PLLC2_OUT0 5 = PLLC3_OUT0 6 = CLK_M
15:8	0x0	VIC_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 VIC_CLK_DIVISOR.
7:0	0x0	VIC_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.7.258 CLK_RST_CONTROLLER_PLLP_OUTC_0

Offset: 0x67c | Read/Write: R/W | Reset: 0x00030000 (0b00000000xxxxx011xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	PLLP_OUT5_RATIO: PLLP_OUT5 divider from base PLLP (lsb denote 0.5x).

Bit	Reset	Description
18	DISABLE	PLLP_OUT5_OVRRIDE: 0 = disallow PLLP_OUT5 ratio override, 1 = enable override. 0 = DISABLE 1 = ENABLE
17	ENABLE	PLLP_OUT5_CLKEN: PLLP_OUT5 divider clk enable. 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
16	RESET_DISABLE	PLLP_OUT5_RSTN: PLLP_OUT5 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

5.7.259 CLK_RST_CONTROLLER_PLLP_MISC1_0

Offset: 0x680 | Read/Write: R/W | Reset: 0x30000000 (0bx011xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	0x0	PLLP_OUT5_DIV_BYP: 1 = bypass PLLP_OUT5 divider.
29	ENABLE	PLLP_HSIO_CLK_EN: Clock gate control for PLLP clock branch to HSIO. 1 = enable, 0 = disable 0 = DISABLE 1 = ENABLE
28	ENABLE	PLLP_XUSB_CLK_EN: Clock gate control for PLLP clock branch to XUSB. 1 = enable, 0 = disable 0 = DISABLE 1 = ENABLE

5.7.260 CLK_RST_CONTROLLER_EMC_DIV_CLK_SHAPER_CTRL_0

EMC_DIV_CLK_SHAPER_CTRL and EMC_PLLC_SHAPER_CTRL shaper control registers should only be changed when the shaper paths are not being selected

EMC_DIV_CLK_SHAPER_CTRL shaper path is selected when ENABLE_EMC_DIV_SHAPER is set and EMC_2X_CLK_SRC is not 111

EMC_PLLC_SHAPER_CTRL shaper is selected when ENABLE_EMC_PLLC_SHAPER is set and EMC_2X_CLK_SRC is 111

Offset: 0x68c | Read/Write: R/W | Reset: 0x00000c00 (0b0xxxxxxxxxxxxxxxxxxx110000000000)

Bit	Reset	Description
31	DISABLE	ENABLE_EMC_DIV_SHAPER: DISABLE: Not using the shaped clock for int_div_clk in the EMC switch 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
11	0x1	EMC_DIV_CLK_SHAPER_N0: Adjust Duty Cycle in step sizes smaller than the delay of the inverter
10	0x1	EMC_DIV_CLK_SHAPER_N1: Adjust Duty Cycle in step sizes smaller than the delay of the inverter
9	0x0	EMC_DIV_CLK_SHAPER_P0: Adjust Duty Cycle in step sizes smaller than the delay of the inverter
8	0x0	EMC_DIV_CLK_SHAPER_P1: Adjust Duty Cycle in step sizes smaller than the delay of the inverter
7	DEFAULT_SETTING	EMC_DIV_CLK_SHAPER_CTRL_TRIM_SELECT: DEFAULT_SETTING: Controlled by shaper pins (RTL): default_values[5:0]. SW_PROG: Controlled by Software programmable register EMC_DIV_CLK_SHAPER_CTRL[5:0] 0 = DEFAULT_SETTING 1 = SW_PROG
6	ACTIVE	EMC_DIV_CLK_SHAPER_CTRL_SW_BYPASS: ACTIVE: Shaper circuit in clock path BYPASS: Shaper Bypassed. S -> Y path to reduce distortion 0 = ACTIVE 1 = BYPASS
5	DISABLE	EMC_DIV_CLK_SHAPER_CTRL_DF: DISABLE: No delay on CLKIN(fall) -> CLKOUT(fall) ENABLE: Delay added on CLKIN(fall) -> CLKOUT(fall) Based on the EMC_DIV_CLK_SHAPER_CTRL[4:3] register 0 = DISABLE 1 = ENABLE
4:3	0x0	EMC_DIV_CLK_SHAPER_CTRL_FALL_DELAY
2	DISABLE	EMC_DIV_CLK_SHAPER_CTRL_DR: DISABLE: No delay on CLKIN(rise) -> CLKOUT(rise) ENABLE: Delay added on CLKIN(rise) -> CLKOUT(rise) Controlled by the EMC_DIV_CLK_SHAPER_CTRL[1:0] register. 0 = DISABLE 1 = ENABLE
1:0	0x0	EMC_DIV_CLK_SHAPER_CTRL_RISE_DELAY

5.7.261 CLK_RST_CONTROLLER_EMC_PLLC_SHAPER_CTRL_0

EMC_DIV_CLK_SHAPER_CTRL and EMC_PLLC_SHAPER_CTRL shaper control registers should only be changed when the shaper paths are not being selected

EMC_DIV_CLK_SHAPER_CTRL shaper path is selected when ENABLE_EMC_DIV_SHAPER is set and EMC_2X_CLK_SRC is not 111

EMC_PLLC_SHAPER_CTRL shaper is selected when ENABLE_EMC_PLLC_SHAPER is set and EMC_2X_CLK_SRC is 111

Offset: 0x690 | Read/Write: R/W | Reset: 0x00000c00 (0b0xxxxxxxxxxxxxxxxxxx110000000000)

Bit	Reset	Description
31	DISABLE	ENABLE_EMC_PLLC_SHAPER: DISABLE: Not using the shaped clock for PLLC in the EMC switch 0 = DISABLE 1 = ENABLE
11	0x1	EMC_PLLC_SHAPER_N0: Adjust Duty Cycle in step sizes smaller than the delay of the inverter
10	0x1	EMC_PLLC_SHAPER_N1: Adjust Duty Cycle in step sizes smaller than the delay of the inverter
9	0x0	EMC_PLLC_SHAPER_P0: Adjust Duty Cycle in step sizes smaller than the delay of the inverter
8	0x0	EMC_PLLC_SHAPER_P1: Adjust Duty Cycle in step sizes smaller than the delay of the inverter
7	DEFAULT_SETTING	EMC_PLLC_SHAPER_CTRL_TRIM_SELECT: DEFAULT_SETTING: Controlled by shaper pins (RTL): default_values[5:0]. SW_PROG: Controlled by Software programmable register EMC_PLLC_SHAPER_CTRL[5:0] 0 = DEFAULT_SETTING 1 = SW_PROG
6	ACTIVE	EMC_PLLC_SHAPER_CTRL_SW_BYPASS: ACTIVE: Shaper circuit in clock path BYPASS: Shaper Bypassed. S->Y path to reduce distortion 0 = ACTIVE 1 = BYPASS
5	DISABLE	EMC_PLLC_SHAPER_CTRL_DF: DISABLE: No delay on CLKIN(fall) -> CLKOUT(fall) ENABLE: Delay added on CLKIN(fall) -> CLKOUT(fall) based on the EMC_PLLC_SHAPER_CTRL[4:3] register 0 = DISABLE 1 = ENABLE
4:3	0x0	EMC_PLLC_SHAPER_CTRL_FALL_DELAY
2	DISABLE	EMC_PLLC_SHAPER_CTRL_DR: DISABLE: No delay on CLKIN(rise) -> CLKOUT(rise) ENABLE: Delay added on CLKIN(rise) -> CLKOUT(rise) Controlled by the EMC_PLLC_SHAPER_CTRL[1:0] register 0 = DISABLE 1 = ENABLE
1:0	0x0	EMC_PLLC_SHAPER_CTRL_RISE_DELAY

6.0 CL-DVFS

The CL_DVFS module contains control logic that provides a hardware mechanism for controlling the clock rate and power supply voltage of the fast CPU complex.

This section describes the closed-loop dynamic voltage and frequency scaling (CL-DVFS) registers. It is not intended to be a programming guide to CL-DVFS, as it is expected that NVIDIA supplied drivers will always be used with it. However, the register interface is documented to aid with understanding those drivers.

6.1 CL-DVFS Registers

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

6.1.1 CL_DVFS_CTRL_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	DFLL_CTRL_MODE: DFLL Control Mode. 0 = DISABLE: Disabled. The ring oscillator does not run. 1 = ENABLE_OPEN_LOOP: In Enable Open Loop mode, the ring oscillator will run, but the control loop will remain inactive. 2 = ENABLE_CLOSED_LOOP: In Enable Closed Loop mode, the control loop is enabled and the I2C interface will continuously update the control output in order to maintain the desired frequency set in the DFLL_FREQ_REQ_xxxx fields.

6.1.2 CL_DVFS_CONFIG_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000006 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000110)

Bit	Reset	Description
7:0	0x6	DFLL_CONFIG_DIV_S: Refclk divider for setting DFLL control loop sample rate. Refclk is divided by 32x the value in this field

6.1.3 CL_DVFS_PARAMS_0

Offset: 0x8 | Read/Write: R/W | Reset: 0x000000f0 (0bxxxxxxx000000000xxxxx00011110000)

Bit	Reset	Description
24	0x0	DFLL_PARAMS_CG_SCALE: If set to 1, reduces the overall loop gain by a factor of 8.
23:22	0x0	DFLL_PARAMS_FORCE_MODE: 0 = DISABLE : Disabled. The I2C control value is never forced during a frequency change. 1 = FIXED : In Fixed Delay mode, the I2C control value is forced for a fixed number of sample periods. 2 = AUTO: In Auto mode, the I2C control value is forced for a calculated number of sample periods.
21:16	0x0	DFLL_PARAMS_CF_PARAM: Length of time that a forced I2C control value will be applied after a frequency change if forcing was requested for that change. In Fixed Delay mode, it provides the number of sample periods to force the I2C control value. In Auto mode, the force time is equal to I2C control output delta * cf_param / 16
10:8	0x0	DFLL_PARAMS_CI_PARAM: Integral term gain in the control loop controls how a cycle deficit/surfeit after a frequency change is cleared. It temporarily raises or lowers the core voltage to allow the total clock cycle count to converge with the ideal number of cycles that should have been produced since the last frequency change request. 0 = DISABLE 1 = DIV2 2 = DIV4 : 3 = DIV8 4 = DIV16

Bit	Reset	Description
		5 = DIV32 : 6 = DIV64 7 = DIV128
7:0	0xf0	DFLL_PARAMS_CG_PARAM: Overall loop gain control (SIGNED value), controls the overall response time of the control loop

6.1.4 CL_DVFS_TUNE0_0

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

Bit	Reset	Description
23:16	0x0	DFLL_TUNE0_DLY_STK: Input bits to both coarse (4) and fine (4) tune the delay
15:8	0x0	DFLL_TUNE0_DLY_SRAM: Input bits to both coarse (3) and fine (5) tune the delay of the SRAM path.
7:0	0x0	DFLL_TUNE0_DLY_INV: Input bits to both coarse (3) and fine (5) tune the delay of the inverter path

6.1.5 CL_DVFS_TUNE1_0

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:11	0x0	DFLL_TUNE1_DLY_FINE: Input bits to tune the two phases of the clock. 8 bits to tune, and 1 bit to choose high vs. low
10:0	0x0	DFLL_TUNE1_DLY_WIRE: Input bits to both coarse (2) and fine (9) tune the delay of wire dominated path

6.1.6 CL_DVFS_FREQ_REQ_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x0000ff40 (0bxxx00000000000011111111101000000)

Bit	Reset	Description
28	0x0	DFLL_FREQ_REQ_FORCE_EN: Set to '1' to force I2C control output to initial value specified by the 'force_val' field
27:16	0x0	DFLL_FREQ_REQ_FORCE_VAL: Value forced onto the integrator during a frequency transition, only used if the 'force_en' field below is '1' AND the global 'force_mode' field of the dfll_params register is not set to 'disable'. The best value is (desired I2C control value - safe I2C control value) x 128) / Cg.
15:8	0xff	DFLL_FREQ_REQ_SCALE: Proportion of output clock cycles (+1) to *not* skip over a period of 256 cycles
7	0x0	DFLL_FREQ_REQ_VALID: Set to '1' to indicate that the frequency configuration is valid and should be used. If this bit is '0', the control loop will revert to 'open loop' mode, and the I2C control interface value will be determined by the 'safe' value.
6:0	0x40	DFLL_FREQ_REQ_MULT: Primary frequency multiplication factor 'F'. The ring oscillator output frequency will be (REF_CLK/2) x F.

6.1.7 CL_DVFS_SCALE_RAMP_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	DFLL_OUTPUT_RAMP_RATE: The ramp up/ramp down rate of the control signal to the output scaler. Determines the number of cycles (+1) to wait for each counter step.

6.1.8 CL_DVFS_DROOP_CTRL_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000810 (0bxxxxxxxx00000000xxxx100000010000)

Bit	Reset	Description
23:16	0x0	DFLL_DROOP_CTRL_MIN_FREQ: The minimum allowed ring oscillator frequency before the 'droop' clock skipper is enabled. The value written determines the minimum number of cycles that are allowed in 4 REF_CLK cycles' Fmin = droop_ctrl.min_freq * (REF_CLK / 4)
11:8	0x8	DFLL_DROOP_CTRL_CUT: CPU clock is scaled by (cut+1)/16 immediately after reaching the minimum ring oscillator frequency
7:0	0x10	DFLL_DROOP_CTRL_RATE: Controls the rate at which clock cycles are re-introduced to the droop skipper after it has been ramped down to compensate for a frequency droop. It is the number of cycles (+1) to wait for each counter step

6.1.9 CL_DVFS_OUTPUT_CFG_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x50200000 (0bx1010000xx100000xx00000000000000)

Bit	Reset	Description
30	0x1	DFLL_OUTPUT_CONFIG_I2C_ENABLE: Master enable control for I2C control value updates. If this field is '0', then I2C control messages are inhibited, regardless of the DFLL mode.
29:24	0x10	DFLL_OUTPUT_CONFIG_SAFE: 'Safe' value for the OUTPUT control interface. This value will be output whenever OUTPUT is enabled but the DFLL is disabled, or in open loop mode.
21:16	0x20	DFLL_OUTPUT_CONFIG_MAX: Maximum allowed value on the OUTPUT control interface.
13:8	0x0	DFLL_OUTPUT_CONFIG_MIN: Minimum allowed value on the OUTPUT control interface.
7	0x0	DFLL_OUTPUT_CONFIG_DELTA_EN: 1: In conjunction with 'clk_en'=1, causes the PWM clock/data output to only become active (for 32 cycles) whenever a change in the PWM value occurs. 0: The PWM data/clock outputs run continuously when enabled.
6	0x0	DFLL_OUTPUT_CONFIG_CLK_EN: Enables the PMIC control clock output for digitally controlled PMICs.
5:0	0x0	DFLL_OUTPUT_CONFIG_DIV_D: Divider setting for PWM PMIC control output (divides the SOC clock).

6.1.10 CL_DVFS_OUTPUT_FORCE_0

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6	0x0	DFLL_OUTPUT_FORCE_ENABLE: Enable the force value onto the OUTPUT control output
5:0	0x0	DFLL_OUTPUT_FORCE_VALUE: Value to force OUTPUT control output whenever the i2c_ctrl_force_enable field is set to 1.

6.1.11 CL_DVFS_MONITOR_CTRL_0

Offset: 0x28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2:0	0x0	DFLL_MONITOR_CTRL_SELECT: Selects a control loop data source for monitoring 0 = DISABLE 1 = CYCLE_INT 2 = PRO_TERM 3 = INT_TERM 4 = OUTPUT_INT 5 = OUTPUT_VALUE 6 = FREQ

6.1.12 CL_DVFS_MONITOR_DATA_0

Offset: 0x2c | Read/Write: RO | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
16	X	DFLL_MONITOR_DATA_NEW: Set to '1' whenever a new sample is generated. Cleared on reads.
15:0	X	DFLL_MONITOR_DATA_VAL: Read data monitor source selected by dfll_monitor_ctrl_select. If monitoring is enabled, this field is updated every sample period.

6.1.13 CL_DVFS_I2C_CFG_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x0010f000 (0bxxxxxxxxxxx1x0001111x00000000000)

Bit	Reset	Description
20	0x1	I2C_BUS_ARB: Enables arbitration for bus
18:16	0x0	I2C_MASTER_CODE: Master code for high-speed transfers
15	0x1	I2C_PACKET_MODE: Enables Packet mode for high-speed transfers
14:12	0x7	I2C_SIZE: Size of voltage values from 1 to 7
10	0x0	I2C_ADDR_7BIT_10BIT: 0: Selects 7-bit addressing 1: Select 10-bit addressing
9:0	0x0	I2C_SLAVE_ID: External slave ID address

6.1.14 CL_DVFS_I2C_VDD_REG_ADDR_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	I2C_DEFAULT_DATA: Default data
7:0	0x0	I2C_ADDR_DATA: Address for voltage sel

6.1.15 CL_DVFS_I2C_STS_0

Offset: 0x48 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
6:1	X	I2C_LAST_VALUE: Output value from the DFLL of last I2C request that was completed successfully
0	X	I2C_REQ_PENDING: Indicates there is an outstanding I2C request

6.1.16 CL_DVFS_INTR_STS_0

Offset: 0x5c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	MAX_VDD_LIMIT_INTR: Interrupt to indicate the DFLL has hit the VDD ceiling as programmed in DFLL_OUTPUT_CONFIG_MAX
0	0x0	MIN_VDD_LIMIT_INTR: Interrupt to indicate the DFLL has hit the VDD floor as programmed in DFLL_OUTPUT_CONFIG_MIN

6.1.17 CL_DVFS_INTR_EN_0

Offset: 0x60 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	MAX_VDD_LIMIT_INTR_EN: Enable for interrupt to indicate the DFLL has hit the VDD ceiling as programmed in DFLL_OUTPUT_CONFIG_MAX
0	0x0	MIN_VDD_LIMIT_INTR_EN: Enable for interrupt to indicate the DFLL has hit the VDD floor as programmed in DFLL_OUTPUT_CONFIG_MIN

6.1.18 CL_DVFS_I2C_CLK_DIVISOR_REGISTER_0

The divisor values (N) must be programmed so that:

- SCL frequency (Std/Fast/Fm+ modes) = ClkSourceFreq/((tlow+thigh+3)*(N+1)) for lower values of N, up to 3
- SCL frequency (Std/Fast/Fm+ modes) = ClkSourceFreq/((tlow+thigh+2)*(N+1)) for higher values of N, above 3
- SCL frequency (HS mode) = ClkSourceFreq/((ths_low+ths_high+4)*(N+1)) for lower values of N, up to 4
- SCL frequency (HS mode) = ClkSourceFreq/ ((ths_low+ths_high+2)*(N+1)) for higher values of N, above 4

Offset: 0x16c | Read/Write: R/W | Reset: 0x00190001 (0b00000000000110010000000000000001)

Bit	Reset	Description
31:16	0x19	I2C_CLK_DIVISOR_STD_FAST_MODE:N = Divide by n+1
15:0	0x1	I2C_CLK_DIVISOR_HSMODE:N = Divide by n+1

6.1.19 CL_DVFS_OUTPUT_LUT_0

This lookup table (LUT) contains voltage LUT values stored in a 33-deep by 8 bit wide RAM. To program the LUT, simply sequentially address the RAM, using the base offset address of 0x20 (10’b10xxxxxxx). Addresses are DWORD aligned, not BYTE aligned. PMIC values must be programmed to span the entire desired operating range, and be monotonically increasing. The logic assumes that the middle entry (16th) corresponds to the ‘safe’ initial voltage, as specified in the DFLL_OUTPUT_CONFIG.SAFE field. DFLL_OUTPUT_CONFIG_SAFE is initialized by default to 0x10, which corresponds to the middle of the LUT. However, the safe value can point to any LUT entry.

[THIS PAGE INTENTIONALLY LEFT BLANK]

7.0 TIMERS

This section documents the various timers available to software in a Tegra® K1 system. The following table summarizes these timers.

> **Note:** The same logic is present in both Tegra K1 variants. References to Shadow CPU, Cluster1, and CPUs 2 and 3 in the Fast CPU (FCPU) cluster are not relevant to Tegra K1 64-bit.

Table 23: List of Timers

Name	Primary Use	Related Interrupt	Secure	Freq.	Notes
RTC	Wall Clock Timer	RTC	Pseudo	32kHz	See the Real-Time Clock section in this TRM for detailed information.
TMR	NVIDIA® Generic Timers	TMR9-0	Cfg	1MHz	Configurable to be secure
WDT	TMR timers used as Watchdog	WDT_<>	Cfg	1MHz	Configurable to be secure
TSC	Reference for GIT	N/A	Yes	OSC	Counter value can only be updated in secure mode.
GIT	ARM CPU Generic Timers	PPIs*	Yes	TSC	These timers use TSC as reference.

*PPIs are per CPU Private Peripheral Interrupts

7.1 ARM CPU Generic Timers (GITs)

The ARM® Generic Timer architecture defines generic CPU timer requirements. The architecture requires that the SoC supply a timestamp counter reference that is independent of the CPU clock frequency.

The CPU Generic Timer includes:

- A physical counter that contains the count value of the system counter.
- A virtual counter that indicates virtual time. The virtual counter contains the value of the physical counter minus a 64-bit virtual offset.
- A set of four timers per CPU.

The four timers are the following:

- Secure Physical Timer
- Non-Secure Physical Timer
- Hypervisor Timer
- Virtual Timer

Refer to the appropriate ARM Architecture Reference Manual for generic timer specifications.

7.2 Generic Timer System Counter (TSC)

The ARM Generic Timer Specification requires a system time-stamp counter (TSC) supplied by the SoC.

The TSC is implemented as part of the Tegra K1 Power Management Controller (PMC). It is a 56-bit counter which runs at the crystal oscillator clock frequency. Refer to the Power Management Controller section in this TRM for additional TSC information.

7.3 NVIDIA Timers (TMR)

The programmable timer block contains ten 29-bit programmable timer counters and one 32-bit timestamp counter.

Timer interval, one-shot, or periodic interrupts are configured in the Timer Present Value register (PTV). When enabled, the timer loads the Timer Present Value count and begins decrementing every one microsecond. The timer generates a timer request when the count reaches zero.

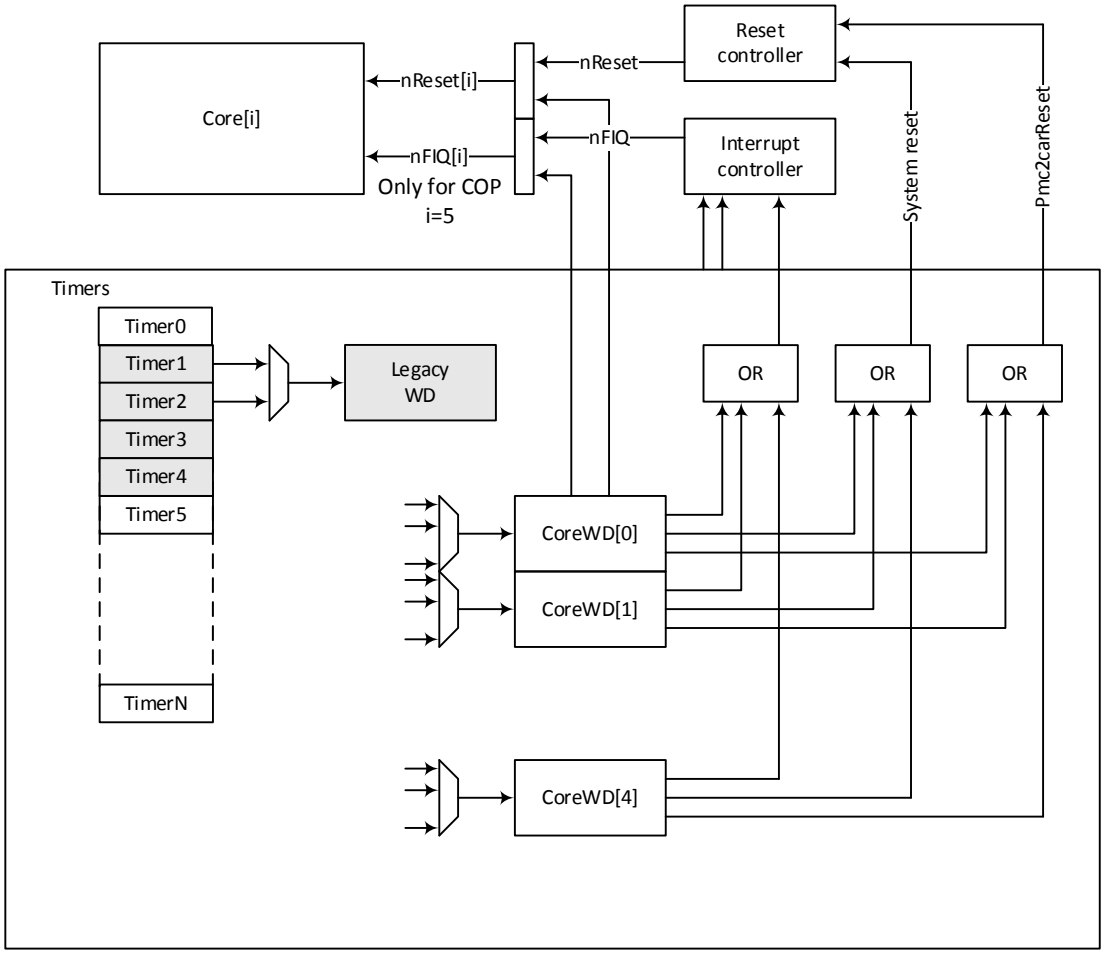
If the PTV periodic control bit is set, the timer reloads the PTV interval and continues decrementing the timer.

The timer count is 29 bits wide, supporting timer intervals of 536.87 seconds maximum or 1 microsecond minimum. Bit 30 of the periodic Timer Value (PTV) is the enable bit for the timer. Reading the PTV Register clears any pending timer interrupt.

A read-only Preset count register (PCR) allows the processor to inspect the current value of the timer decrement counter. The timestamp counter (32 bits wide) is cleared to zero on reset, and counts upwards every 1 µs.

The figure below shows a reference decomposition of the timers in functional blocks.

Figure 13: Timer Functional Block Diagram



- The timers are numbered 0 to 9
- All timers have a dedicated interrupt bit
- Timers 0, 6, 7, 8, and 9 also share a common interrupt called SharedTimerInterrupt.
- Timer interrupts are assigned as follows:
  - Timer 1 = primary interrupt controller bit 0

- Timer 2 = primary interrupt controller bit 1
- Timer 3 = secondary interrupt controller bit 9
- Timer 4 = secondary interrupt controller bit 10
- Timer 5 = fourth interrupt controller bit 25
- Timer 6 = fifth interrupt controller bit 24
- Timer 7 = fifth interrupt controller bit 25
- Timer 8 = fifth interrupt controller bit 26
- Timer 9 = fifth interrupt controller bit 27
- Timer 0 = fifth interrupt controller bit 28
- SharedTimerInterrupt = hierarchical group 1, bit 10

7.4 Watchdog Timers (WDTs)

A watchdog timer facilitates recovery from system lockup conditions. The watchdog timer can interrupt the processor when the selected timer counter expires. Under normal operation, this interrupt is serviced by reading the timer status register (TIMER_WDTx_STATUS). Any timer in the Tegra K1 timer block may be configured as a watchdog timer.

- The Tegra K1 series processor provides five Watchdog Timers, one each for main CPUs and one for COP (AVP): WDT0 is allocated to CPU0 of cluster0 or the shadow CPU of cluster1
- WDT1 is allocated to CPU1
- WDT2 is allocated to CPU2
- WDT3 is allocated to CPU3
- WDT4 is allocated to COP (AVP)

The 5 watchdog timers are directly associated with a processor core, i.e., 4 CPU cores and the COP:

- The watchdog is reset by the corresponding core reset signal
- WD timers 0 to 3 are associated with CPU 0 to 3, WD 4 is associated with COP

There are no dedicated timers for the WDTs. Each WDT can be configured to use any one of the TMRs as its timer source.

Based on the configuration of the TIMER_WDTx_CONFIG register, a WDT works as follows:

- When the timer decrements to zero the first time, a WDT interrupt can be generated
- When the timer consecutively decrements down to zero a second time, without the processor reading the interrupt status register (TIMER_TMRx_TMR_PCR), a per core FIQ can be generated. This feature only works for the COP; the corresponding signal does not propagate to the CPU cores.
- When the timer consecutively decrements down to zero a third time, without the processor reading the interrupt status register (TIMER_TMRx_TMR_PCR), a per CPU reset can be generated.
- When the timer consecutively decrements down to zero a fourth time, a system reset can be generated. The cause of this system reset can be read by reading PMC_RST_STATUS[RST_SOURCE] (see the Power Management Controller section for more information).

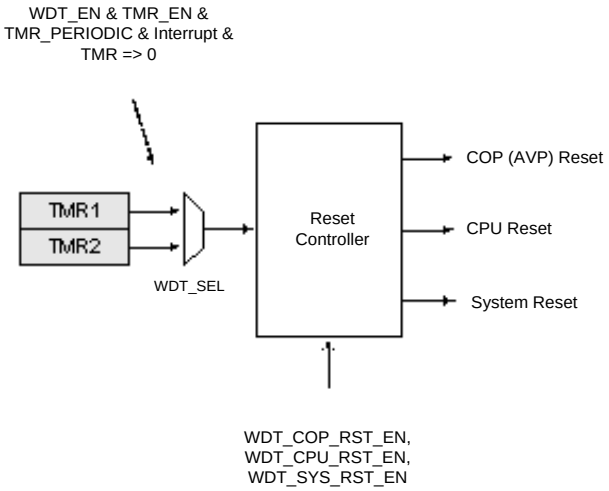
7.5 Secure TMRs and Secure WDTs

The Tegra K1 series processor TMRs and/or WDTs can be configured to be secure timers. As described in TIMER_SECURE_CFG register table later in this section, a secure register TIMER_SECURE_CFG -- which itself can only be written by secure writes – can be configured such that each of the TMRs and WDTs can be independently configured to be secure. By default all of TMRs and WDTs are non-secure and backward compatible.

7.6 Legacy Watchdog Timer

The legacy watchdog timer can use one TMR1 or TMR2 as the source and can generate a set of reset signals. This facility is deprecated and replaced by the more versatile WDTs.

Figure 14: Watchdog Block Diagram



The legacy watchdog timer can be programmed to reset just the CPU, just the AVP, or the entire system.

If only the CPU is reset, then the AVP will continue running as-is. The CPU reset vector should be programmed in advance so that it skips the iROM boot code and jumps directly into its own CPU boot routine (which can be anywhere).

If only the AVP is reset, the same is true. The CPU keeps running as-is. If the AVP reset vector is set to default, it runs the normal boot code. It is up to the Boot Loader to decide if it needs to restart the CPU as well.

The watchdog timer can be programmed to the maximum timeout value of 0x3fffffff. At 1 µs/count, this is more than one thousand seconds.

The watchdog timer operates as follows:

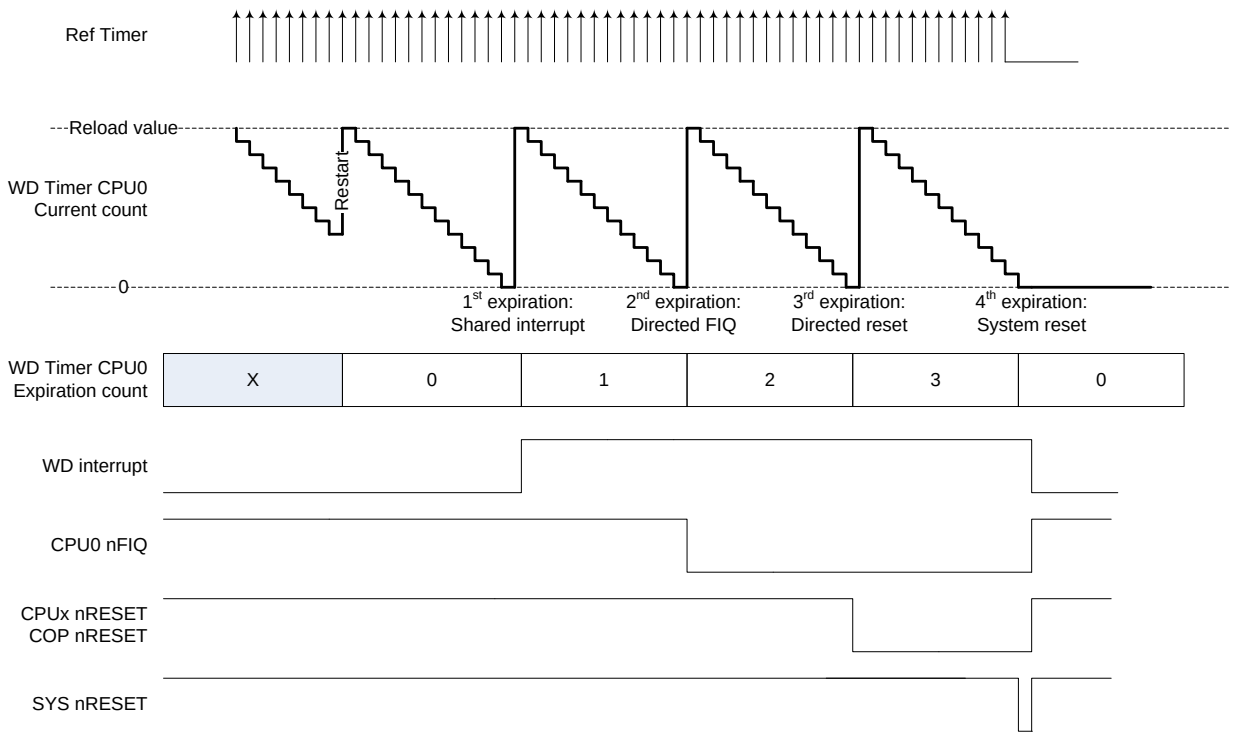
1. Watchdog generates an interrupt based on the Timer
    a. TMR1 or TMR2 can be selected as source
    b. Timer should be programmed as Periodic with Interrupt.
    c. CAR.RST_SOURCE.WDT_EN = 1
    d. CAR.RST_SOURCE.WDT_SEL selects TMR2/TMR1
2. Reset can be for the AVP, CPU, or full system
    a. CAR.RST_SOURCE.WDT_***_RST_EN = 1
3. Upon reboot, CAR.RST_SOURCE.WDT_*_RST_STA indicates the cause of the reset

7.7 Watchdog Timer Programming Guide

The five watchdog timers operate as follows:

- Select any one of the general-purpose timers as timing reference
- Standard down counter with programmable period, that also counts expirations, specific action are taken at the counter expiration
  - If the expiration count is 0, a normal interrupt may be generated (probably pooled)
  - If the expiration count is 1, the FIQ for the corresponding processor may be asserted, this operation bypasses the normal interrupt controller
  - If the expiration count is 2, the reset for a programmable set of processors is asserted, this may be the null set
  - If the expiration count is 3, a system wide reset may be asserted (two variants, standard system or closer to external hardware reset)

Figure 15: 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.

7.8 Timers Registers

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

Each timer uses two registers called PTV and PCR at two consecutive word addresses. The format is the same for all 10 timers and is defined only once with <t> taking values between 0 and 9. Note that the start offset sequence is haphazard for legacy reasons.

Table 24: Start Offsets for PTV and PCR Timer Registers

Timer	Start Offset for Register Pair
TMR1	0x00
TMR2	0x08
TMR3	0x50
TMR4	0x58
TMR5	0x60
TMR6	0x68
TMR7	0x70
TMR8	0x78
TMR9	0x80
TMR0	0x88

7.8.1 TIMER_TMR<t>_TMR_PTV_0

Parameter <t> takes a value from 0 to 9. The start offset for the timer is defined relative to the start address in Table 24.

Before programming the timer:

- Input oscillator frequency must be specified in the TIMERUS_USEC_CFG register.
- Program the number of 1 µs timer counts per "tick" in the PTV (Present Trigger Value) register.
- Set the PTV EN (enable bit) to start counting down.
- When the count reaches zero, an interrupt is generated.
- To auto-reload the timer counter, set the PTV PER (periodic) bit. Otherwise, leave it clear for a one-shot.

Timer Present Trigger Value (Set) Register

Offset: See Table 24 │ Read/Write: R/W │ Reset: 0x00000000 (0b00x00000000000000000000000000000)

Bit	Reset	Description
31	0x0	EN: Enable Timer 0 = DISABLE 1 = ENABLE
30	0x0	PER: Enable Periodic Interrupt 0 = DISABLE 1 = ENABLE
28:0	0x0	TMR_PTV: Trigger Value: count trigger value (count length). This is in n+1 scheme. If you program the value n, the count trigger value will actually be n+1.

7.8.2 TIMER_TMR<t>_TMR_PCR_0

Parameter <t> takes a value from 0 to 9. The start offset for the timer is defined relative to the start address in Table 24.

When an interrupt is generated, write "1" to PCR[30] to clear the interrupt

Timer Present Count Value (Status) Register

Offset: See Table 24 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
30	RW	0x0	INTR_CLR: 1 = clears the interrupt, 0 = no affect. Write-1-to-Clear
28:0	RO	X	TMR_PCV: Counter value: decrements from PTV.

7.9 Fixed Time Base Registers

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

The USEC_CFG/CNTR_1US registers provide a fixed time base (in microseconds) to be used by the rest of the system regardless of the clk_m frequency (i.e., 12 MHz, 13 MHz, 19.2 MHz, 26 MHz, or other frequencies).

7.9.1 TIMERUS_CNTR_1US_0

This free-running read-only register/counter changes once every microsecond and is used mainly by hardware (can also be used by software). It starts counting from 0 once it is out of system reset and will continue counting forever, unless the oscillator clock is stopped or during a system reset.

(A) Software Use

Although there is no interrupt mechanism for this register, software can read the content of this register multiple times to determine the amount of time that has elapsed.

(B) Hardware Use

- Used by the 4 timers to determine whether the programmed timer value has been reached; these 4 timers can trigger interrupts.
- To provide a periodic USEC pulse to be used by the flow controller to count the programmable number of microseconds before a flow control condition is triggered.
- Used by secure boot logic.

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

Bit	Reset	Description
31:16	x	HIGH_VALUE: Elapsed time in microseconds
15:0	x	LOW_VALUE: Elapsed time in microseconds

7.9.2 TIMERUS_USEC_CFG_0

Software should first configure this register by telling what fraction of 1 microsecond each clk_m represents. For example, if the clk_m is running at 12 MHz, then each clk_m represents 1/12 of a microsecond.

"USEC_DIVIDEND" and "USEC_DIVISOR" are used to indicate what fraction of 1 microsecond each clk_m represents.

Table 25: clk_m Frequency Indicator for USEC

clk_m 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

Offset: 0x4 | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxxxxxxxx0000000000001100) | Default: 0x0000000b

Bit	Reset	SW Default	Description
15:8	0x0	NONE	USEC_DIVIDEND: Microsecond dividend. (n+1)
7:0	0xc	0xb	USEC_DIVISOR: Microsecond divisor. (n+1)

7.9.3 TIMERUS_CNTR_FREEZE_0

Offset: 0x3c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

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.

7.10 Watchdog Timers

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

Each Watchdog timer uses four registers called CONFIG, STATUS, COMMAND, and UNLOCK PATTERN, at four consecutive word addresses. The format is the same for all four timers and is defined only once with <w> taking values from 0 to 4.

The following register offsets are relative to the TMR block base.

Each Watchdog timer has the following:

- CoreWatchdog Configuration Register
- Timer sources are numbered 1 through 10 (0 is not used and results in undefined behavior)

- Period of 0 results in MAX period
- The generated interrupt is dependent on the WD index.
- WD0 to 3 generate a CPU_WD_Interrupt,
- WD4 generates a COP_WD_Interrupt
- CPU0 settings apply to both clusters
- Core numbering: 0-3 applies to CPU0-CPU3, 4 applies to COP

7.10.1 TIMER_WDT<w>_CONFIG_0

Parameter <w> takes a value from 0 to 4.

Core Watchdog Configuration Register

Offset: 0x100 + (0x20 * <w>) | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
20:16	0x0	CoreResetBitmapEn: Enable reset of a set of cores at third expiration counter (one bit per processor) same level the signal from PMC asserted in case of external reset.
15	0x0	Pmc2CarResetEn: Enable Full system reset at fourth expiration of the counter 0 = DISABLE 1 = ENABLE
14	0x0	SystemResetEnable: Enable system reset at fourth expiration of the counter, at same level as existing watchdog interrupt 0 = DISABLE 1 = ENABLE
13	0x0	FIQEnable: Enable FIQ assertion at second expiration of the counter 0 = DISABLE 1 = ENABLE
12	0x0	InterruptEnable: Enable interrupt at first expiration of the counter 0 = DISABLE 1 = ENABLE
11:4	0x0	Period: Measured in periods of selected timer, this is the reload value
3:0	0x0	TimerSource: Select the timer used as reference (TMR0 – TMR9) 0 = TMR0 1 = TMR1 2 = TMR2 3 = TMR3 4 = TMR4 5 = TMR5 6 = TMR6 7 = TMR7 8 = TMR8 9 = TMR9

7.10.2 TIMER_WDT<w>_STATUS_0

Parameter <w> takes a value from 0 to 4.

Core Watchdog Status

Offset: 0x104 + (0x20 * <w>) | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
13:12	X	CurrentExpirationCount: Current count of expiration since the last start of expiration
11:4	X	CurrentCount: Current value of the counter
2	X	FiqStatus: Current status of FIQ, cleared by start
1	X	InterruptStatus: Current status of interrupt, cleared by start
0	X	Enabled: 1 when counter is active. When true, writes to the corresponding config register are ignored

7.10.3 TIMER_WDT<w>_COMMAND_0

The StartCounter bit enables watchdog counter operation, loads the watchdog counter, starts the watchdog timer to count down, resets the expiration count to 0, and clears all flags. Also used as restart.

The counter can be disabled by setting the DisableCounter bit, but only if the unlock register has been programmed before with the correct pattern. Writing to the command register always clears the disable unlock register. When set while StartCounter is 0 and the unlock register contains the unlock pattern the Watchdog transitions back to disabled.

The register fields are Write to set only.

Parameter <w> takes a value from 0 to 4.

CoreWatchdogCommand Register

Offset: 0x108 + (0x20 * <w>) | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	DisableCounter: Write 1 to disable WDT counter if unlocked
0	0x0	StartCounter: Write 1 to enable WDT counter

7.10.4 TIMER_WDT<w>_UNLOCK_PATTERN_0

Parameter <w> takes a value from 0 to 4.

CoreWatchdogDisableUnlock Register

Offset: 0x10c + (0x20 * <w>) | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	UnlockPattern: Must be written with value 0xC45A to allow write to DisableCounter to take effect. The pattern is reset at each write to Command Reg.

7.11 Timer Shared Interrupt Status

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

TimerSrcBitmap is used for Timers 6 through 10 which share a common interrupt line to the controller. Bit b is set if timer 6+b generated an interrupt. The corresponding bit is cleared by writing INTR_CLR bit of the TMR_PCR register.

WatchdogSrcBitmap is used for the 5 watchdog timers. This is the set of Interrupt Status bits from the corresponding CoreWatchdogStatus, but as a bitmap.

7.11.1 SHARED_INTR_STATUS_0

Offset: 0x0 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:6	X	TimerSrcBitmap: Timer[0,9:6] interrupt status in bitmap form
4:0	X	WatchdogSrcBitmap: WDT[4:0] interrupt status in bitmap form

7.11.2 SHARED_TIMER_SECURE_CFG_0

Offset: 0x4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxx0xxx00000xx0000000000)

Bit	Reset	Description
20	0x0	USEC: TIMERUS_USEC_CFG secure mode config. If this bit is set (=1) then TIMERUS_USEC_CFG register can only be written by secure writes. If this bit is cleared then TIMERUS_USEC_CFG register can be written by secure or non-secure writes. This register can always be read by secure or non-secure reads. Note: this bit should be set if any one of the TMR is enabled to be secure. 1: ENABLE 0: DISABLE
16	0x0	WDT4: WDT4 secure mode config. See the WDT0 bit description. This bit is reserved since WDT4 belongs to COP and it has no notion of Secure or Non-Secure transactions
15	0x0	WDT3: WDT3 secure mode config. See the WDT0 bit description.
14	0x0	WDT2: WDT2 secure mode config. See the WDT0 bit description.
13	0x0	WDT1: WDT1 secure mode config. See the WDT0 bit description.
12	0x0	WDT0: WDT0 secure mode config. If this bit is set (=1) then WDT0 registers (i.e. TIMER_WDT0_{CONFIG,COMMAND,UNLOCK_PATTERN}) can only be written by secure writes. A non-secure read will not trigger read-side-effect, but would return proper read value. If this bit is cleared then the WDT0 registers can be written by secure or non-secure writes. WDT0 registers can always be read by secure or non-secure reads. Note: if this bit is set, then the TMR used for this WDT0 should be enabled to be secure. 1: ENABLE 0: DISABLE
9	0x0	TMR9: Timer9 secure mode config. See the TMR0 bit description.
8	0x0	TMR8: Timer8 secure mode config. See the TMR0 bit description.
7	0x0	TMR7: Timer7 secure mode config. See the TMR0 bit description.
6	0x0	TMR6: Timer6 secure mode config. See the TMR0 bit description.
5	0x0	TMR5: Timer5 secure mode config. See the TMR0 bit description.
4	0x0	TMR4: Timer4 secure mode config. See the TMR0 bit description.
3	0x0	TMR3: Timer3 secure mode config. See the TMR0 bit description.

Bit	Reset	Description
2	0x0	TMR2: Timer2 secure mode config. See the TMR0 bit description.
1	0x0	TMR1: Timer1 secure mode config. See the TMR0 bit description.
0	0x0	TMR0: Timer0 secure mode config. If this bit is set (=1) then TMR0 registers (i.e., TIMER_TMR0_TMR_{PTV,PCR}) can only be written by secure writes. A non-secure read will not trigger read-side-effect, but would return proper read value. If this bit is cleared then TMR0 registers can be written by secure or non-secure writes. TMR0 registers can always be read by secure or non-secure reads. 0: DISABLE 1: ENABLE

8.0 MULTI-PURPOSE I/O PINS AND PIN MULTIPLEXING (PINMUXING)

8.1 Overview

Tegra® K1 devices can be configured with different I/O functions on particular pins to allow use in a variety of different configurations. This section discusses how this is controlled and how the pins themselves are set up.

Many of the pins on Tegra K1 devices are connected to multi-purpose I/O (MPIO) pads. An MPIO can operate in two modes: either acting as a signal for a particular I/O controller, referred to as a Special-Function I/O (SFIO); or as a software-controlled general-purpose I/O function, referred to as GPIO. Each MPIO has up to four SFIO functions as well as being a GPIO.

Though each MPIO has up to 5 functions (a GPIO function and up to 4 SFIO functions), a given MPIO can only act as a single function at a given point in time. The Pinmux controller in Tegra K1 devices includes the logic and registers to select a particular function for each MPIO.

This section describes the following features of MPIOs, the Pinmux controller, and the GPIO controller:

- Basic capabilities of the MPIO pads
- Differences between the variety of MPIO pads
- Mapping of MPIO pads to I/O controllers (i.e., pinmuxing)
- Behavior of the MPIO pads during power-up
- Behavior of the MPIO pads before, during, and after deep sleep
- Recommendations for software programming related to the MPIO pads.

This section covers the Multi-Purpose digital I/O pads. It does not address the special-purpose I/O pads, such as those used for USB, IC_USB, SATA, PCIE, TVO/DAC, MIPI DSI, MIPI CSI, the oscillator, or DRAM interfaces.

8.2 Terms and Acronyms

Some common terms used with Tegra K1 processor pinmuxing are as follows:

Term	Definition
CZ	Controlled-output impedance MPIO pads
DD	Dual-driver MPIO pads
Deep sleep	Also known as LP0. A full-chip power state in which VDD_CORE is turned off but VDD_RTC (i.e., VDD_AO) and many of the I/O power rails remain on
DPD	Deep Power Down (a mode in which the pad can tolerate VDD_CORE being turned off)
GPIO	General-Purpose I/O
LV	Low-voltage MPIO pads
MPIO	Multi-Purpose I/O
OD	Open-drain MPIO pads
SFIO	Special-Function I/O
ST	Standard MPIO pads

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:
  - Schmitt mode, CMOS mode, or both
- A weak pull-up and a weak pull-down

Tegra K1 devices include five types of MPIO pads, which all share a common structure. The following table summarizes the differences between the five MPIO pad types.

Table 26: MPIO Pad Types

Pad Type	I/O Rail Voltage	Input Buffer	Output Buffer	I/O Voltage Tolerance	Nominal Pull Strength	"Slew Rate" Control	Drive Strength Control
ST	1.8, 2.8-3.3	Schmitt & CMOS	push-pull	VDDIO	100 kΩ	2 bits, up & down	5 bits, up & down
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
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, up

The ST (standard) MPIO pads are the most common pads on the chip.

The DD (dual-driver) MPIO pads are similar to the ST pads with the addition of a 3.3V tolerant true open-drain mode. A DD pad can tolerate its I/O pin being pulled up to 3.3V (regardless of supply voltage) as long as the pad's output-driver is set to open-drain mode. There are special power-sequencing considerations when using this functionality.

Note: Refer to "Power Sequencing" in the *NVIDIA Tegra K1 Processors Data Sheet* (DS-06742-001) for a complete description of the power-up sequencing requirements for Tegra K1 processors.
Refer to the same section in the *NVIDIA Tegra K1 64-Bit Processors Data Sheet* (DS-07070-001) for Tegra K1 64-bit processors.

The OD (open drain) MPIO pads are optimized to tolerate 5V on the I/O pin regardless of the supply voltage. They are similar to ST pads except for an improved I/O voltage tolerance, the absence of a weak pull-up, and the absence of a push-pull output driver.

The CZ (controlled output impedance) MPIO pads are optimized for use in applications requiring a tightly controlled output impedance. They are similar to ST pads except for changes in the drive strength circuitry and in the weak pull-ups/pull-downs. Tegra K1 processors include CZ pads on the VDDIO_SDMMC1 and VDDIO_SDDMC3 power rails. Each of those rails also includes a pair of CZ_COMP pads. Circuitry within the Tegra K1 device continually matches the output impedance of the CZ pads to the on-board pull-up/-down resistors attached to the CZ_COMP pads.

The LV (low voltage) MPIO pads are optimized for use with a 1.2V supply voltage (and signaling level). They support a 1.8V supply voltage (and signaling level) as a secondary mode. The Tegra K1 processors include LV pads on VDDIO_SDMMC4. The SDMMC4 interface also has a pair of COMP pads, i.e., LV_COMP pads, to generate impedance code.

Note: Refer to "Pin Definitions" in the *NVIDIA Tegra K1 Processors Data Sheet* (DS-06742-001) for the list of the pad types and the nominal pull-up/-down strength associated with each MPIO. See the columns labeled "MPIO Pad Type" and "Nominal Pull Strength." The power rail information is located in the "Power Sequencing" section.
Refer to the same section in the *NVIDIA Tegra K1 64-Bit Processors Data Sheet* (DS-07070-001) for Tegra K1 64-bit processors.

8.4 Pad Controls

The Tegra K1 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.

8.4.1 Per Pad Options

The following controls can be independently configured on a per-pad basis:

PUPD	Internal Pull-up/down option: Option to enable internal Pull-up, Pull-down resistors or neither
TRISTATE	Tristate (high-z) option: Disables or enables the pad's output driver. This setting overrides any other functional setting and also whether pad is selected for SFIO or GPIO. Can be used when the pad direction changes or the pad is assigned to different SFIO to avoid glitches.
E_INPUT	Input Receiver (Enable/Disable): Enables or disables input receiver.
OD	Open Drain option: (Applies to DD pads only) Selects between an open-drain output driver and a push-pull driver.
IO_RESET	Not used.
RCV_SEL	(Applies to OD pads only). Selects between "High VIL/VIH" and "Normal VIL/VIH" receivers. RCV_SEL=1: "High VIL/VIH" RCV_SEL=0: "Normal VIL/VIH"

During normal operation, these per-pad controls are driven by the pinmux controller registers. See the section called "Pinmuxing" below for more information.

During deep sleep, the PMC bypasses and then resets the pinmux controller registers. Software should reprogram these registers as necessary after returning from deep sleep. See the section called "Deep Sleep Behaviors" for more information on the interaction of PMC, software, and the pinmux controller following deep sleep.

8.4.2 Per Pad Control Group

The MPIO pads are partitioned into 35 "pad control groups". The following controls can be configured independently for each pad control group. Pad control groups are associated with a group of pads that share similar functionality for example, GMI, SDMMC.

HSM	High Speed Mode (Enable/Disable)
SCHMT	Schmitt Trigger (Enable/Disable)
LPMD	Low Power Mode. Select Low Power Modes (different impedance/current values)
DRVDN / UP	Drive Down / Up. Driver Output Pull-Up/Pull-Down drive strength code. Normally, the code is 5 bits but for CZ type pads, it is 7 bits.
SLWR/ SLWF	Slew Falling / Rising. Driver Output Pull-Up/Pull-Down slew control code. Normally, the code is 2 bits. In general, the code corresponding to the minimum slew is set (2'b11).

The controls are configured via the "pad control group registers". There is one pad control register per pad control group. During deep sleep, all of these pad control registers automatically return to their power-on-reset state. Software should reprogram these registers as necessary following deep sleep.

Table 27 lists the register address for each pad control group. Additionally, the table describes the bit positions of the controls within each register.

For example, writing a 1 to bit 3 of register 0x700000884 will enable the Schmitt trigger mode for the pads in group "cdev1cfg" pins, DAP_MCLK1 and DAM_MCLK1_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 27: Pad Control Groups Register Addresses

Pad Control Group	Register Address	PREEMP	HSM	SCHMT	DRV_TYPE	DRVDN	DRVUP	SLWR	SLWF
gmacfg	0x70000900	0	2	3	7:6	18:14	24:20	29:28	31:30
sdio1cfg	0x700008ec		2	3		18:12	26:20	29:28	31:30
sdio3cfg	0x700008b0		2	3		18:12	26:20	29:28	31:30
sdio4cfg	0x700009c4		2	3		16:12	24:20	29:28	31:30
aocfg0	0x700009b0		2	3		16:12	24:20	29:28	31:30
aocfg1	0x70000868		2	3		16:12	24:20	29:28	31:30
aocfg2	0x7000086c		2	3		16:12	24:20	29:28	31:30
aocfg3	0x700009a8		2	3		16:12		29:28	
aocfg4	0x700009c8		2	3	7:6	18:12	26:20	29:28	31:30
cdev1cfg	0x70000884		2	3		16:12	24:20	29:28	31:30
cdev2cfg	0x70000888		2	3		16:12	24:20	29:28	31:30
ceccfg	0x70000938		2	3		16:12	24:20	29:28	31:30
dap1cfg	0x70000890		2	3		16:12	24:20	29:28	31:30
dap2cfg	0x70000894		2	3		16:12	24:20	29:28	31:30
dap3cfg	0x70000898		2	3		16:12	24:20	29:28	31:30
dap4cfg	0x7000089c		2	3		16:12	24:20	29:28	31:30
dap5cfg	0x70000998		2	3		16:12	24:20	29:28	31:30
dbgcfg	0x700008a0		2	3		16:12	24:20	29:28	31:30
ddccfg	0x700008fc		2	3		16:12	24:20	29:28	31:30
dev3cfg	0x7000092c		2	3		16:12	24:20	29:28	31:30
owrcfg	0x70000920		2	3		16:12	24:20	29:28	31:30
spicfg	0x700008b4		2	3		16:12	24:20	29:28	31:30
uaacfg	0x700008b8		2	3		16:12	24:20	29:28	31:30
uabcfg	0x700008bc		2	3		16:12	24:20	29:28	31:30
uart2cfg	0x700008c0		2	3		16:12	24:20	29:28	31:30
uart3cfg	0x700008c4		2	3		16:12	24:20	29:28	31:30
udacfg	0x70000924		2	3		16:12	24:20	29:28	31:30
atcfg1	0x70000870		2	3	7:6	18:12	26:20	29:28	31:30
atcfg2	0x70000874		2	3	7:6	18:12	26:20	29:28	31:30
atcfg3	0x70000878		2	3	7:6	18:12	26:20	29:28	31:30
atcfg4	0x7000087c		2	3	7:6	18:12	26:20	29:28	31:30
atcfg5	0x70000880		2	3		18:14	23:19	29:28	31:30
atcfg6	0x70000994		2	3	7:6	18:12	26:20	29:28	31:30
gmecfg	0x70000910		2	3		18:14	23:19	29:28	31:30
gmfcfg	0x70000914		2	3		18:14	23:19	29:28	31:30
gmgcfg	0x70000918		2	3		18:14	23:19	29:28	31:30
gmhcfg	0x7000091c		2	3		18:14	23:19	29:28	31:30
hvcfg0	0x700009b4		2	3		16:12		29:28	
gpvcfg	0x70000928		2	3		16:12	24:20	29:28	31:30
usb_vbus_en_cfg	0x7000099c		2	3		16:12	24:20	29:28	31:30

8.4.3 Per I/O Power Rail

These per-power-rail controls are included in the PMC registers which maintain their state during deep sleep. Software does NOT need to reprogram these register following deep sleep. The following table summarizes the functionality of these signals along with how they are controlled in the PMC.

Pin Name	Pin Description	Control/Grouping Mechanism in PMC
E_33V	Active high 3.3V mode select. When low, selects VDDP 1.8V mode.	Generated based on respective PWR_DET signal and the logic is maintained in the PMC, i.e., AO domain By default, it is maintained at Logic 1 to ensure safe power up of I/Os that are pulled to 3.3V
E_NO_IOPWR	Active high. When high, prevents leakage	Maintained per power rails for 16 power rails. Ideally, should have

Pin Name	Pin Description	Control/Grouping Mechanism in PMC
	when I/O power is gone while core power is still on.	been set during Cold Reset so that I/O rail voltage ramping does not affect the pad. However, it is reset in the chips. Whenever an I/O power rail is shut off on any specific use case, this bit has to be set to Logic1 before the I/O rails are brought down to avoid excessive leakage to the pad.

Nine pads are used for Power Detect status of various I/O rails. They are used to sense the I/O rail voltage level (3.3V or not). They are reflected through the PWR_DET set of registers in the PMC which will in turn control the E_33V pin of the pad.

8.5 Pinmuxing

Tegra K1 processors include many types of I/O controllers (such as I^2C, SDMMC, NAND, and GMI). For some of these controller types, Tegra K1 processors include multiple "controller instances". For example, Tegra K1 processors include five active I^2C controller instances.

Each controller instance on a Tegra K1 processor communicates with external devices via a set of "external signals". For each controller instance, each signal can be brought out on (at least) one MPIO. For example, each I^2C controller has two external signals: CLK and DAT. The CLK signal of the I2C1 controller is available on the MPIO whose ball name is GEN1_I2C_SCL.

Many of the pins on Tegra K1 devices are connected to multi-purpose I/O (MPIO) pads. To connect an MPIO to a particular I/O controller, configure it as a Special-Function I/O (SFIO) rather than a General-Purpose I/O (GPIO). Each MPIO has up to four SFIO functions. Each MPIO can as also be programmed to function as a GPIO. For example, the pin named UART3_CTS_can act in one of these five ways:

- As a GPIO
- As the signal CTS for UART3
- As the signal CMD for SDMMC
- As the signal CLK for DTV
- As the signal CS3 for SPI4B controller

Though each MPIO supports multiple functions, a given MPIO can only act as a single function at a given point in time. The Pinmux controller on a Tegra K1 device includes the logic and registers to select a particular function for each MPIO.

Some controller instances have a particular signal available on more than one MPIO. Before using any controller, make sure that the pinmux registers are programmed to bring each signal out on at most ONE MPIO. For example, UART3's TXD signal is available on the MPIOs whose ball names are ULPI_CLK, GMI_A16, etc. In a system which brings out UART3's TXD signal on the GMI_A16 ball, the pinmux registers for ULPI_CLK should be programmed to select some other signal.

Some controller instances make their entire set of signals available on two or more sets of MPIOs. That is, such controllers have more than one "interface". Before using any controller, make sure that the pinmux registers are programmed to bring out the controller's signals on at most ONE interface.

Note: The Pinmux controller includes one register per MPIO. It can be controlled on a per-pin basis.

Figure 16 shows the pinmux logic associated with a single MPIO. The ENB in the diagram (equivalent to the EN pin at the pad) is active Low. Thus, the pad is active when ENB is Low and is in the High-Z state when ENB is High.

In the figure below, GPIO_OE and SFIO_OE[3:0] are considered active Low; that is, when they are Low, the pad's output is enabled.

Figure 16: Structure of Pinmux Controller (per MPIO)

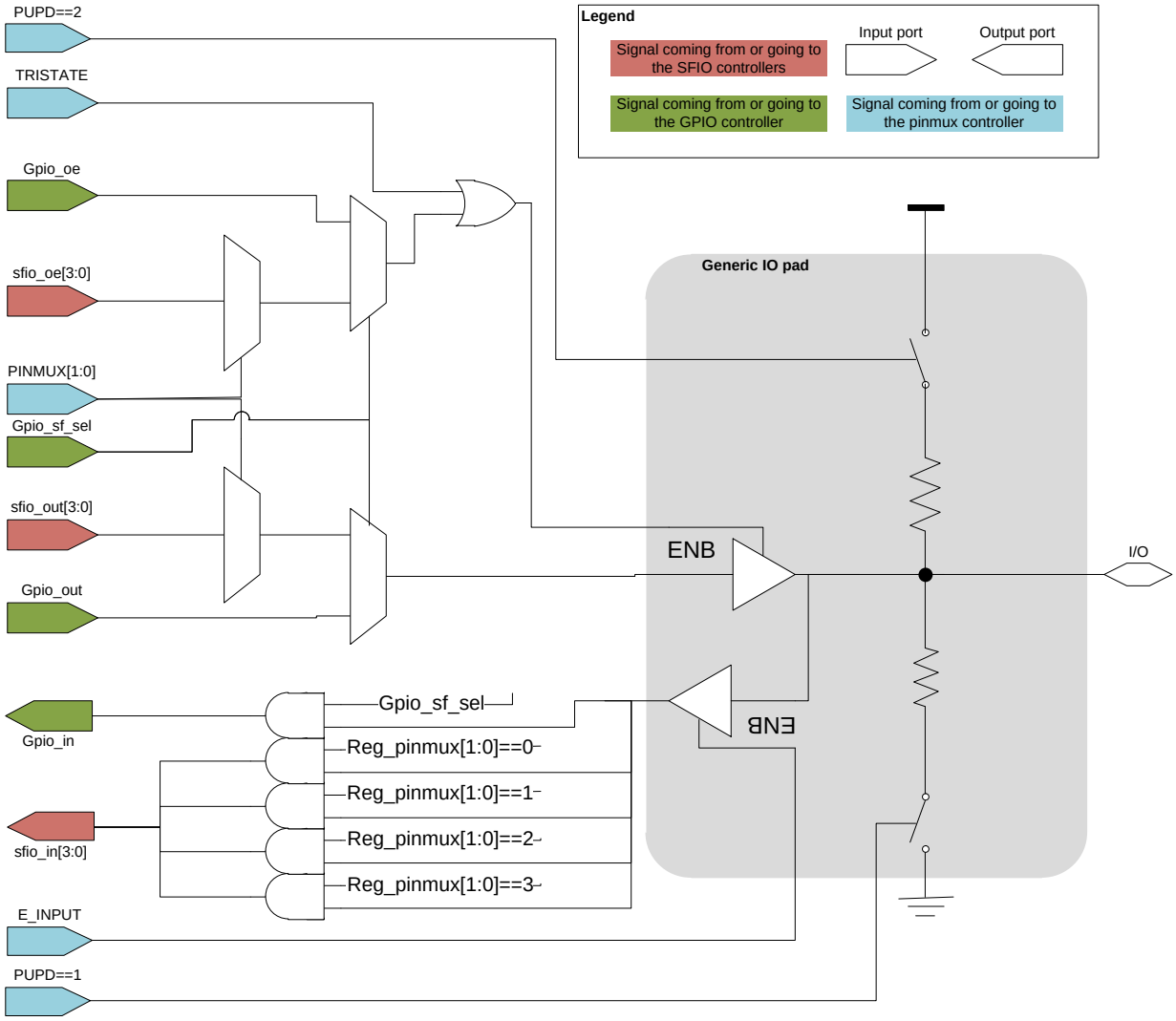


Table 28 describes the layout of each Pinmux controller register.

Table 28: Pinmux Register Format

Field Name	Bit Position	Default Value	Description
RCV_SEL	9	0/1 (depends on the type of interface)	Applicable for "OD" pads
IO_RESET	8	-	This is a dummy pin for MPIO pads and not used. The effect of applying IO_RESET is achieved by keeping A and EN pins of the pads to Logic 0.

Field Name	Bit Position	Default Value	Description
LOCK	7	0	0: Writes to this register are accepted 1: Writes to this register are ignored (until the next wake from Deep Sleep) This is a sticky bit. Once software sets this bit to 1, the only way to clear it is to reset the chip or enter and exit Deep Sleep.
E_OD	6	DD pads:1	0: the pad's output driver operates in push-pull mode. 1: the pad's output driver operates in open-drain mode. WARNING: the DD pads are only 3.3V tolerant when this bit is set. When this bit is cleared, the voltage tolerance is limited to the I/O power supply voltage. Before pulling/driving external I/Os to a value greater than the I/O power supply, this bit must be set to avoid excessive leakage. Default value recommended for DD pads is 1 to ensure that during power up no excessive leakage occurs irrespective of I/O voltage. Weak pull-ups are disabled if E_OD is set to 1 for DD pads.
		Other pads: 0	This bit only matters for DD pads. It is not implemented for other MPIO pad types.
E_INPUT	5	0/1 (depends on the type of functionality)	0: the pad's input receiver is disabled. 1: the pad's input receiver is enabled. Refer to the Tegra K1 Pinmux spreadsheet.
TRISTATE	4	0/1 (depends on the type of functionality)	0: the pad's output driver is enabled. 1: the pad's output driver is disabled (i.e., the output driver is in a high-impedance state). Refer to the Tegra K1 Pinmux spreadsheet.
PUPD	3:2	0-2 (depends on the type of functionality)	0: the pad's weak pull-up and pull-down are both disabled 1: the pad's weak pull-down is enabled and the weak pull-up is disabled. 2: the pad's weak pull-up is enabled and the weak pull-down is disabled. 3: this is an illegal combination. Refer to the Tegra K1 Pinmux spreadsheet.
PINMUX	1:0	00	0: set the pinmux logic for SFIO0. 1: set the pinmux logic for SFIO1. 2: set the pinmux logic for SFIO2. 3: set the pinmux logic for SFIO3. Note that the choice between SFIO and GPIO is controlled via the GPIO registers. Refer to the Tegra K1 Pinmux spreadsheet.

Refer to the Tegra K1 Pinmux spreadsheet for a list of the SFIO functions supported by each MPIO on Tegra K1 devices. Each SFIO is listed in the table with the following format: <ControllerType><ControllerInstance><InterfaceLetter>-<Signal>

- **ControllerType** indicates the type of I/O controller associated with this SFIO function
- **ControllerInstance** indicates the I/O controller, if the Tegra K1 processor has more than one I/O controller of the specified type
- **InterfaceLetter** differentiates between the pin sets, if the controller instance can connect to more than one set of pins
- **Signal** identifies the particular role this SFIO plays for the I/O controller.

For some of the MPIOs, the hardware includes SFIO functions that are omitted from the spreadsheet. Many of these are deprecated. Contact NVIDIA for more information.

8.6 Cold Boot Reset

The following list is a simplified description of the device power on/boot process for Tegra K1 devices concentrating on those aspects which relate to the MPIO pins.

- System-level hardware executes the power-up sequence. This sequence ends when system-level hardware releases SYS_RESET_N.
- Each MPIO pin has a deterministic power-on-reset state. The particular reset state for each pin is chosen to minimize the need of on-board components like pull-up resistors in a Tegra K1 based system. For example, the on-chip weak pull-ups are enabled during power-on-reset for pins which are usually used to drive active-low chip selects.
- Pins that act as a strap source for cold boot must have their inputs enabled (through E_INPUT control). If they have MPIO Pinmuxing enabled, the strap source should be the primary function.
- The rising edge of SYS_RESET_N does not trigger any of the MPIOs to change their output state.

> **Note:** Refer to "POR Behavior" in the *NVIDIA Tegra K1 Processors Data Sheet* (DS-06742-001) for a list of the power-on-reset states for each of the MPIOs.
> Refer to the same section in the *NVIDIA Tegra K1 64-Bit Processors Data Sheet* (DS-07070-001) for Tegra K1 64-bit processors.

Following the power-up sequence, most of the MPIOs on the chip will stay in their power-on-reset state until system-dependent software (e.g., the Boot Loader or the OS) reprograms them. However, depending on the secondary boot device used in a given system, the Boot ROM may change the state of some of the MPIOs. The following sections list the MPIOs that the Boot ROM uses for each type of secondary boot device.

8.7 Secondary Boot

The following list is a simplified description of the device boot process for Tegra K1 devices concentrating on those aspects which relate to the MPIO pins.

1. System-level hardware executes the power-up sequence. This sequence ends when system-level hardware releases SYS_RESET_N.
2. The Boot ROM on the Tegra K1 device begins executing and programs the on-chip I/O controllers to access the secondary boot device.
3. The Boot ROM on the Tegra K1 device fetches the BCT and Boot Loader from the secondary boot device.
4. If the BCT and Boot Loader are fetched successfully, Boot ROM on the Tegra K1 device yields to the Boot Loader.
5. Otherwise, Boot ROM on the Tegra K1 device enters USB recovery mode.

Each MPIO pin has a deterministic power-on-reset state. The particular reset state for each pin is chosen to minimize the need of on-board components like pull-up resistors in a Tegra K1 based system. For example, the on-chip weak pull-ups are enabled during power-on-reset for pins which are usually used to drive active-low chip selects.

Pins that act as a strap source must have their inputs enabled by default. If they have MPIO functionality, then the default configuration group should choose strap functionality.

> **Note:** Refer to " POR Behavior" in the *NVIDIA Tegra K1 Processors Data Sheet* (DS-06742-001) for a list of the power-on-reset states for each of the MPIOs.
> Refer to the same section in the *NVIDIA Tegra K1 64-Bit Processors Data Sheet* (DS-07070-001) for Tegra K1 64-bit processors.

System designs must adhere to the described power-up sequence.

Following the power-up sequence, most MPIOs on the Tegra K1 device will stay in their power-on-reset state until system-dependent software (the Boot Loader or the OS) reprograms them. However, depending on the secondary boot device used in a given system, the Boot ROM may change the state of some of the MPIOs and associated pinmux so that the Boot ROM can interface with the secondary device to fetch the Boot Loader.

The following subsections list the MPIOs that the Boot ROM uses for each type of secondary boot device. Depending on the type of secondary boot device used, the associated I/O rails have to be brought up by the Boot ROM/PMIC to facilitate the data exchange.

8.7.1 SPI Boot

SPI4C is used for a SPI interface for secondary boot. The pins are spread across different MPIOs as summarized below.

Ball Name	SFIO Usage	I/O Power Rail	Default Pinmux Selection	Change Needed?
GPIO_PI3	SPI4C_CS0	vddio_gmi	ALTERNATE2	YES. To be changed to ALTERNATE3
GPIO_PG5	SPI4C_SCK	vddio_gmi	ALTERNATE2	YES. To be changed to ALTERNATE3
GPIO_PG7	SPI4C_DIN	vddio_gmi	ALTERNATE2	YES. To be changed to ALTERNATE3
GPIO_PG6	SPI4C_DOUT	vddio_gmi	ALTERNATE2	YES. To be changed to ALTERNATE3

TRISTATE control of all the pins needs to be changed to NORMAL. Also the pinmux selection should be changed to select ALTERNATE3.

8.7.2 eMMC Boot

There are four SDMMC interfaces; the one used as a secondary boot interface is SDMMC4. SDMMC4 has all the necessary functionality for SDMMC assigned as primary function, and thus there is no need to change the Pinmux configuration. However, the Boot ROM has to remove the TRISTATE control bit on these pins (through the Pinmux control register) to ensure the pins do not have the overriding TRISTATE that is making the pins inputs.

Ball names include:

- SDMMC4_CLK
- SDMMC4_CMD
- SDMMC4_DAT0
- SDMMC4_DAT1
- SDMMC4_DAT2
- SDMMC4_DAT3
- SDMMC4_DAT4
- SDMMC4_DAT5
- SDMMC4_DAT6
- SDMMC4_DAT7

8.8 Deep Sleep Behaviors

Deep Sleep is an ultra-low-power standby state in which a Tegra K1 device maintains much of its I/O state while most of the chip is powered off. The following lists offer a simplified description of the deep sleep entry and exit concentrating on those aspects which relate to the MPIO pads.

8.8.1 Deep Sleep Entry

The steps to enter deep sleep are as follows:

1. LP-entry software programs the PMC to alert about Deep Sleep entry.

2. LP-entry software also programs the APBDEV_PMC_IO_DPD_REQ registers to emulate a software based overriding request to enter the DPD. This is done to control the deassertion of E_DPD during LP0 exit which is needed to arrest glitches associated with I/Os on the LP0 sequence.
3. I/O controllers are brought into the IDLE state.
4. The PMC latches much of Tegra K1's I/O state, i.e., currently driven by functional logic by sampling the pinmux controllers output. There are small variations on this logic depending on the pad type (mentioned below). Latching of the I/O state is triggered by writing into APBDEV_PMC_DPD_SAMPLE_0 register maintained in the PMC register space.
5. Along with latching the I/O values, various pad controls, such as E_IO_HV,E_33V, have to be latched and driven depending on the pad type.
6. The PMC runs the state machine to drive the control pins in the pad that puts them in the DPD state. There are two control pins (E_DPD and SEL_DPD) which have to be driven with proper timing as per the Pad data sheet. The PMC logic provides a configurable timer to select this. The timer is provided in the APBDEV_PMC_SEL_DPD_TIM_0 register. Entering into DPD is triggered through APBDEV_PMC_IO_DPD_REQ_0. Both registers are in the PMC register space.
7. The PMC deasserts CORE_PWR_REQ.
8. The PMIC powers down VDD_CORE.
9. The PMC continues to drive the I/O state that it has latched
10. As a result of the LP0 entry sequence, the pads are put in the DPD state by asserting E_DPD and SEL_DPD. When these signals are asserted, the core supply to the pad (VAUXC_CORE) can be successfully removed and the pad can be powered using VDD_AO/VDD_RTC. This way the pads keep the I/O states yet consume low power.

> **Note:** For more information about deep sleep entry, refer to the *NVIDIA Tegra K1 Processors Data Sheet* (DS-06742-001) .or the *NVIDIA Tegra K1 64-Bit Processors Data Sheet* (DS-07070-001) as well as the PMC section in this TRM.

8.8.2 Deep Sleep Exit

1. The PMC detects a wake event (for example, a rising edge on an appropriately configured MPIO)
2. The PMC asserts CORE_PWR_REQ.
3. The PMIC powers up VDD_CORE.
4. The PMC resets the logic within VDD_CORE.
5. The Boot ROM executes Warm Boot code, and it does not clear the APBDEV_PMC_DPD_SAMPLE_0 register.
6. The LP0-exit code in DRAM retrieves pinmux registers for every pin and ensures the TRISTATE/PARKED bit is set to 1 in the respective pinmux register.
7. The LP-exit code in DRAM deasserts APBDEV_PMC_IO_DPD_REQ so that the PMC removes E_DPD.
8. The PMC takes the I/Os out of E_DPD but continues to drive them in SEL_DPD mode. This way the I/O values do not change during Deep Sleep exit until platform-specific software (i.e., typically customer owned) takes control of the I/Os.
9. The LP0-exit code in DRAM re-initializes the I/O controllers.
10. The LP0-exit code in DRAM changes the Pinmux configurations to the appropriate controller (based on what was configured during LP0) and clears TRISTATE/PARKED in the Pinmux register of each pin.
11. The LP0-exit code in DRAM, after restoring all the Pinmux register, clears the APBDEV_PMC_DPD_SAMPLE_0 register, which results in SEL_DPD removal of the pad through hardware logic.

> **Note:** For more information about deep sleep exits, refer to the *NVIDIA Tegra K1 Processors Data Sheet* (DS-06742-001) .or the *NVIDIA Tegra K1 64-Bit Processors Data Sheet* (DS-07070-001) as well as the PMC section in this TRM.

8.8.3 Pad Capabilities During Deep Sleep

The MPIO pads do not all have identical behavior during deep sleep. They differ with respect to:

- Output buffer behavior during deep sleep:
  - Does it maintain a programmable (0, 1, or tristate) constant value OR
  - Is it capable of changing state while the chip is still in deep sleep (for example, a pin related to the keyboard controller)?
- Input buffer behavior during deep sleep:
  - Is it forcibly disabled OR
  - Can it be enabled for use as a “GPIO wake event” OR
  - Can it be enabled for some other purpose (for example, a “clock request” pin)?
- Weak pull-up/pull-down behavior during deep sleep
  - Are they forcibly disabled OR
  - Can they be configured?
- Behavior coming out of deep sleep
  - All the pads should maintain the state until software configures the I/O controller
- Pads that do not enter DPD mode during LP0
  - Some pads whose outputs are dynamic have a special type and they don’t enter DPD mode during LP0. The PMC has a configurable option to exclude specific pads from entering DPD during LP0 (see the PMC register APBDEV_PMC_DPD_PADS_ORIDE_0)
  - Pads that are associated with PMIC logic do not enter DPD mode during LP0
  - Some pads like JTAG do not enter DPD mode.

Refer to the “Deep Sleep Behavior” section in the *NVIDIA Tegra K1 Processors Data Sheet* (DS-06742-001) or the *NVIDIA Tegra K1 64-Bit Processors Data Sheet* (DS-07070-001) for a summary of the capabilities of each MPIO during deep sleep.

8.8.3.1 Output Buffers During Deep Sleep

The output buffers of most MPIO pads are “configurable” during deep sleep. Immediately prior to entering deep sleep, the PMC latches the state of the output buffers for such pads. If the pad was driving high (or low) prior to entering deep sleep, the PMC ensures that it continues to drive high (or low) throughout deep sleep. If the pad was tristated during deep sleep, the PMC ensures that it remains tristated throughout deep sleep.

The weak pull-ups and pull-downs of most of the MPIO pads are “disabled” or “not available” during deep sleep to reduce the Tegra K1 device’s static current consumption. For the remainder of the MPIO pads, the weak pull-up and pull-down are “configurable” during deep sleep. If the pull-up (or pull-down) was enabled prior to entering deep sleep, the PMC ensures that it remains enabled throughout deep sleep. Similarly, if the pull-up (or pull-down) was disabled prior to entering deep sleep, the PMC ensures that it remains enabled throughout deep sleep.

Some MPIO pads have an output buffer which behaves in a “special” way during deep sleep. The output state of these pads may change while the Tegra K1 device remains in deep sleep. For example, the keyboard controller may continue scanning the keypad matrix during deep sleep.

8.8.3.2 Input Buffers During Deep Sleep

The input buffers of most MPIO pads are “disabled” during deep sleep. The input buffers are placed in an ultra-low power state. The Tegra K1 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, SYS_CLK_REQ can function as a clock request pin even during deep sleep.

For a complete description of the behavior of these pads during deep sleep, refer to the *NVIDIA Tegra K1 Processors Data Sheet* (DS-06742-001) or the *NVIDIA Tegra K1 64-Bit Processors Data Sheet* (DS-07070-001). Pads whose input can be sampled during Deep Sleep are mentioned as ENABLED in the input buffer column. To ensure the PMC wake logic can sense these wake events, the signals must be directly taken from the ZI pin of the pad because pinmux controls are not available during LP0. Pads that are acting as wake source should satisfy the following condition: The pad’s power rail should be active.

8.8.3.3 Behavior During Deep Sleep Exit

MPIO pads “hold” their deep sleep state until system-dependent software deasserts the TRISTATE bit in the associated Pinmux controller registers. This gives software the opportunity to re-initialize the pinmux and I/O controllers without any glitches in the state of the MPIO pads.

8.9 GPIO Controller

The GPIO controller for Tegra K1 devices provides the tools for configuring each MPIO for use as a software-controlled GPIO.

The GPIO controller is divided into 8 banks. Each bank handles the GPIO functionality for up to 32 MPIOs. Within a bank, the GPIOs are arranged as four ports of 8 bits each. The ports are labeled consecutively from A through Z and then AA through FF. Ports A through D are in bank 0. Ports E through H are in bank 1. In total, there are 162 GPIOs, and the banking and numbering conventions will have some break in between but will maintain backward compatibility in register configurations for the GPIOs as that of previous generation chips.

> **Note:** Refer to the “Pin Descriptions” section in *NVIDIA Tegra K1 Processors Data Sheet* (DS-06742-001) *or the NVIDIA Tegra K1 64-Bit Processors Data Sheet* (DS-07070-001) for a map of each of the Tegra K1 device MPIO pads to a particular GPIO port and bit. See the column labeled “GPIO.”

Each GPIO can be individually configurable as Output/Input/Interrupt sources with level/edge controls.

GPIO configuration has a lock bit controlling every bit separately. When the LOCK bit is set, the associated control aspects of the bits (for example, whether it is an Output/Input or used as GPIO or SFIO or values driven) cannot be modified (locked). The LOCK bit gets cleared only by system reset; it is sticky. This bit can be used for security-related functionality where an authorized entity owning the GPIO can set the configuration and lock it. The lock bit also covers the GPIO output value, so this may not be varied dynamically once lock is enabled.

The GPIO controller also has masked-write registers. Values written to these registers specify both a mask of bits to be updated in the underlying state (the mask bits are not sticky) as well as new values for that state. Individual bits of the state can be updated without the need for a read-modify-write sequence. Thus different portions of software can modify the GPIO controller state without coordination.

8.10 Programming Considerations

8.10.1 Controller Instances with Multiple Pin-outs

Several of the I/O controller instances on a Tegra K1 device have more than one pin-out. That is, the controller can communicate with the outside world via more than one set of pins. Special care is warranted when programming the Pinmux controller for these SFIOs. For a given signal on a given controller’s interface, only a single pin-out should be selected in the pinmux registers.

For example, TXD is an output signal from UART A. It is available on six pins: ULPI_DATA0, SDMMC1_DATA2, SDMMC3_DATA1, GPIO_PU0, GPIO_PS1, and UART2_RTS_N. Depending on the value programmed into the corresponding pinmux registers, each of those six pins might toggle when UART A transmits data.

Similarly, RXD is an input signal to UART A. It is available on six pins: ULPI_DATA1, SDMMC1_DATA1, SDMMC3_CMD, GPIO_PU1, GPIO_PS2, and UART2_CTS_N. Depending on the value programmed into the corresponding pinmux registers, UART A might see the logical-OR of the signal on those six pins. This scenario will almost certainly lead to data corruption. If the UART A RXD signal is brought out on ULPI_DATA1, the pinmux registers for the other five pins should be programmed to some other function.

8.10.2 Resume From Deep Sleep

As the Tegra K1 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.

8.10.3 Unused Pins

For each unused MPIO, assert its tristate and disable its input buffer. For pins whose internal pull-up is enabled during power-on-reset, assert the internal pull-up. Otherwise, assert the internal pull-down.

If all of the pins in a pad-control group are unused, set the drive strengths and slew rates to minimum.

If all of the pins on a power-rail are unused, assert E_NOIOPOWER for that rail in the PMC registers.

8.10.4 Security Considerations for the Pinmux Register

Though attacks via Pinmux are mostly Denial Of Service (DoS) attacks, there are specific cases which are a concern, for example:

- Pins interfacing with Platform PMICs (by reprogramming those pinmuxes, one can cause severe user disruption)
- Any I/O device that is transferring secret data (typically like Serial PROMs having keys). By reprogramming the pinmux, it is possible to redirect the data to wrong interface

In such cases, it is better to make use of the LOCK bit in the Pinmux control register so that values cannot be tampered with. The Boot Loader can perform this task so that customer software does not have to configure those pins. As part of LP0 exit, the lock bit will have to be re-enabled, because the lock is not retained across LP0. Similarly, the GPIO controller has a lock bit for each GPIO to preserve the configuration, value driven, and also the pin is configured as GPIO or SFIO. The configuration can be protected using the lock bit in the GPIO register. This way secure software such as a Boot Loader can ensure that certain pins have the required configuration, irrespective of the actions of less secure software.

8.10.5 Drive Strengths

For all pads that do not have corresponding calibration pads, the drive strength has to be programmed. The common default value assumed for the pad drive impedance is 50 ohms and hence pads have to be programmed with a code that can give 50 ohm impedance. Because the code depends on VDDP, i.e., I/O rail, and also PVT (Process, Voltage and Temperature) corners default a code corresponding to TT-NV-NT (Typical process, Normal Voltage, and Normal Temperature) corner as a function of VDDP voltage is chosen.

In Tegra K1 devices, the primary boot device (SDMMC4) has an auto-calibration option. Thus those pads get an impedance code that matches the PVT conditions, etc. and hence enable the boot process. For all other MPIO pads (which are not involved during boot), the BCT can have a value of the code based on the I/O rail and pad data sheet. The same is programmed by the Boot Loader or any platform-specific software, and it overrides the default Power on Reset values. If the values have to be changed based on SI results, then the same can be loaded in the primary device.

8.11 GPIO Registers

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

Each port of each GPIO controller has several registers for the control and monitoring of the port’s 8 GPIO pins. The registers provide per-pin control of the following features:

- GPIO_CNF_* / GPIO_MSK_CNF — Select SPIO or GPIO
- GPIO_OE_* / GPIO_MSK_OE_* — Output Enable
- GPIO_OUT_* / GPIO_MSK_OUT_* — GPIO Output Value
- GPIO_IN_* — GPIO Input Value (Read Only)
- GPIO_INT_STA_* / GPIO_MSK_INT_STA_* — GPIO Interrupt Status
- GPIO_INT_ENB_* / GPIO_MSK_INT_ENB_* — Interrupt Enable
- GPIO_INT_LVL_* / GPIO_MSK_INT_LVL_* — Interrupt Selection (Edge/Level)
- GPIO_INT_CLR_* — Interrupt Flag Set-to-Clear

Many of the control registers are accessible from two offsets. Accesses to the lower offsets affect all 8 pins for the GPIO port. Accesses to the upper offsets provide a per-pin mask capability allowing adjustments to a subset of the pins. Using the upper offsets can eliminate the need for Read-Modify-Write operations. For example, a write to GPIO_MSK_CNF can substitute for a Read-Modify-Write of GPIO_CNF.

Table 29: Tegra K1 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

Register Name	Offset (Lower: Read-Modify-Write)				Offset (Upper: Per-Pin Mask Write)			
GPIO_OUT_2	220	224	228	22C	2A0	2A4	2A8	2AC
GPIO_IN_2	230	234	238	23C				
GPIO_INT_STA_2	240	244	248	24C	2C0	2C4	2C8	2CC
GPIO_INT_ENB_2	250	254	258	25C	2D0	2D4	2D8	2DC
GPIO_INT_LVL_2	260	264	268	26C	2E0	2E4	2E8	2EC
GPIO_INT_CLR_2	270	274	278	27C				
GPIO Controller 4 – Port	**M**	**N**	**O**	**P**	**M**	**N**	**O**	**P**
GPIO_CNF_3	300	304	308	30C	380	384	388	38C
GPIO_OE_3	310	314	318	31C	390	394	398	39C
GPIO_OUT_3	320	324	328	32C	3A0	3A4	3A8	3AC
GPIO_IN_3	330	334	338	33C				
GPIO_INT_STA_3	340	344	348	34C	3C0	3C4	3C8	3CC
GPIO_INT_ENB_3	350	354	358	35C	3D0	3D4	3D8	3DC
GPIO_INT_LVL_3	360	364	368	36C	3E0	3E4	3E8	3EC
GPIO_INT_CLR_3	370	374	378	37C				
GPIO Controller 5 – Port	**Q**	**R**	**S**	**T**	**Q**	**R**	**S**	**T**
GPIO_CNF_4	400	404	408	40C	480	484	488	48C
GPIO_OE_4	410	414	418	41C	490	494	498	49C
GPIO_OUT_4	420	424	428	42C	4A0	4A4	4A8	4AC
GPIO_IN_4	430	434	438	43C				
GPIO_INT_STA_4	440	444	448	44C	4C0	4C4	4C8	4CC
GPIO_INT_ENB_4	450	454	458	45C	4D0	4D4	4D8	4DC
GPIO_INT_LVL_4	460	464	468	46C	4E0	4E4	4E8	4EC
GPIO_INT_CLR_4	470	474	478	47C				
GPIO Controller 6 – Port	**U**	**V**	**W**	**X**	**U**	**V**	**W**	**X**
GPIO_CNF_5	500	504	508	50C	580	584	588	58C
GPIO_OE_5	510	514	518	51C	590	594	598	59C
GPIO_OUT_5	520	524	528	52C	5A0	5A4	5A8	5AC
GPIO_IN_5	530	534	538	53C				
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

Register Name	Offset (Lower: Read-Modify-Write)				Offset (Upper: Per-Pin Mask Write)			
GPIO_INT_CLR_6	670	674	678	67C				
GPIO Controller 8 – Port	**CC**	**DD**	**EE**		**CC**	**DD**	**EE**	
GPIO_CNF_7	700	704	708		780	784	788	
GPIO_OE_7	710	714	718		790	794	798	
GPIO_OUT_7	720	724	728		7A0	7A4	7A8	
GPIO_IN_7	730	734	738					
GPIO_INT_STA_7	740	744	748		7C0	7C4	7C8	
GPIO_INT_ENB_7	750	754	758		7D0	7D4	7D8	
GPIO_INT_LVL_7	760	764	768		7E0	7E4	7E8	
GPIO_INT_CLR_7	770	774	778					

8.11.1 GPIO_CNF_0

Designate whether each pin operates as a GPIO or as an SFIO. By default all pins come up in SFIO mode. These can be programmed to GPIO mode at any stage. This is an array of 4 identical register entries; the register fields below apply to each entry.

Lock bits are used to control the access to the CNF and OE registers. When set, no one can write to the CNF and OE bits. They can be programmed ONLY during Boot and get reset by chip reset only.

GPIO Port Configuration Registers

Offset: 0h..fh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15	DISABLE	LOCK_7: Lock access to pin 7 CNF, OE and OUT 0 = DISABLE 1 = ENABLE
14	DISABLE	LOCK_6: Lock access to pin 6 CNF, OE and OUT 0 = DISABLE 1 = ENABLE
13	DISABLE	LOCK_5: Lock access to pin 5 CNF, OE and OUT 0 = DISABLE 1 = ENABLE
12	DISABLE	LOCK_4: Lock access to pin 4 CNF, OE and OUT 0 = DISABLE 1 = ENABLE
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

8.11.2 GPIO_OE_0

GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid.

The set of registers below are used to either drive the signal out or as an Input. This needs to be programmed depending upon whether the pin needs to be in either Input or Output.

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port Enable Registers

Offset: 10h..1fh | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = TRI_STATE 1 = DRIVEN
6	0x0	BIT_6: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
5	0x0	BIT_5: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
4	0x0	BIT_4: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
3	0x0	BIT_3: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
2	0x0	BIT_2: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN

Bit	Reset	Description
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

8.11.3 GPIO_OUT_0

GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be valid. This register will take affect only in GPIO mode. This register is used to drive the value out on a given pin.

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port Output Value Registers (Read/Write)

Offset: 20h..2fh | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH
6	0x0	BIT_6: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH
5	0x0	BIT_5: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH
4	0x0	BIT_4: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH
3	0x0	BIT_3: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH
2	0x0	BIT_2: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH
1	0x0	BIT_1: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH
0	0x0	BIT_0: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH

8.11.4 GPIO_IN_0

GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. This is a read-only register used to read the value from the pin. This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port Input Value Registers (Read Only)

Offset: 30h..3fh | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = LOW 1 = HIGH
6	0x0	BIT_6: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH
5	0x0	BIT_5: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH
4	0x0	BIT_4: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH
3	0x0	BIT_3: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH
2	0x0	BIT_2: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH
1	0x0	BIT_1: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH
0	0x0	BIT_0: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH

All GPIO inputs can be independently programmed to generate an interrupt request.

In addition, the individual trigger level for interrupt on each input pin can be programmed as either active-on-high or active-on-low. For example, to program an active-on-high interrupt on bit 3 of GPIO-PORT_C, write '1' into bit 3 of GPIO_INT.LVL.C register (this sets the interrupt to be active-on-high), and then write '1' into bit 3 of GPIO_INT.ENB.C (this enables interrupt on the named bit).

The interrupt flag status can be read in the appropriate bit of the GPIO_INT.STA.C register. Once the programmed interrupt occurs, status should be cleared by writing into the appropriate bit of the GPIO_INT.CLR.C register. Note that the interrupt thus generated is routed to the processor only if the corresponding bit for GPIO interrupts in the Secondary interrupt controller is enabled.

8.11.5 GPIO_INT_STA_0

GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. Every GPIO pin generates an Interrupt when switching from Low-High to High-Low. Interrupt status for each port is saved in an Interrupt status register.

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port Interrupt Status Registers

Offset: 40h..4fh | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE = ACTIVE

Bit	Reset	Description
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

8.11.6 GPIO_INT_ENB_0

Every bit of the GPIO pin has an enable which, when enabled, routes the Interrupt to the Interrupt controller. This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port Interrupt Enable Registers

Offset: 50h..5fh | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
6	0x0	BIT_6: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
5	0x0	BIT_5: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
4	0x0	BIT_4: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
3	0x0	BIT_3: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
2	0x0	BIT_2: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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

8.11.7 GPIO_INT_LVL_0

The GPIO can detect an interrupt for any edge- or level-sensitive signal.

This is an array of 4 identical register entries; the register fields below apply to each entry

GPIO Port Interrupt Activation Level Registers

Offset: 60h..6fh | Read/Write: R/W | Reset: 0b00000000000000000000000000

Bit	Reset	Description
23	0x0	DELTA_7: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
22	0x0	DELTA_6: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
21	0x0	DELTA_5: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
20	0x0	DELTA_4: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
19	0x0	DELTA_3: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
18	0x0	DELTA_2: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
17	0x0	DELTA_1: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
16	0x0	DELTA_0: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
15	0x0	EDGE_7: 1 means Configure as Edge-Triggered Interrupt
14	0x0	EDGE_6: 1 means Configure as Edge-Triggered Interrupt
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

Bit	Reset	Description
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

8.11.8 GPIO_INT_CLR_0

This write-only register clears the Interrupts that are set. This is valid only in GPIO mode when GPIO_INT.ENB is set.

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port Interrupt Flag Set-to-Clear Registers

Offset: 70h..7fh | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
6	0x0	BIT_6: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
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

Bit	Reset	Description
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

8.11.9 GPIO_MSK_CNF_0

Each register is provided with an individual 16-bit version for enabling Masked Writes to avoid a Read-Modify-Write operation by the firmware. The exception is for the interrupt clear register, whose functionality is combined in the interrupt status register. Individual pins only can be programmed by suitably enabling the write masks in the upper byte of these 16-bit registers.

This is an array of 4 identical register entries; the register fields below apply to each entry.

MASKED PRIMARY GPIO/SFIO Config Registers (Masked Writes)

Offset: 80h..8fh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
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

Bit	R/W	Reset	Description
3	RW	0x0	BIT_3: 0 = SPIO 1 = GPIO
2	RW	0x0	BIT_2: 0 = SPIO 1 = GPIO
1	RW	0x0	BIT_1: 0 = SPIO 1 = GPIO
0	RW	0x0	BIT_0: 0 = SPIO 1 = GPIO

8.11.10 GPIO_MSK_OE_0

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Masked Output Enable (Masked Writes)

Offset: 90h..9fh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
7	RW	0x0	BIT_7: 0 = TRI_STATE 1 = DRIVEN
6	RW	0x0	BIT_6: 0 = TRI_STATE 1 = DRIVEN
5	RW	0x0	BIT_5: 0 = TRI_STATE 1 = DRIVEN

Bit	R/W	Reset	Description
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

8.11.11 GPIO_MSK_OUT_0

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO A-D Masked Output Enable (Masked Writes)

Offset: a0h..afh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
7	RW	0x0	BIT_7: 0 = LOW 1 = HIGH
6	RW	0x0	BIT_6: 0 = LOW 1 = HIGH

Bit	R/W	Reset	Description
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

8.11.12 GPIO_MSK_INT_STA_0

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO A-D Masked Interrupt Status (Masked Clears)

Offset: c0h..cfh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
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

Bit	R/W	Reset	Description
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

8.11.13 GPIO_MSK_INT_ENB_0

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO A-D Masked Interrupt Enable (Masked Writes)

Offset: d0h..dfh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
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

Bit	R/W	Reset	Description
7	RW	0x0	BIT_7: 0 = DISABLE 1 = ENABLE
6	RW	0x0	BIT_6: 0 = DISABLE 1 = ENABLE
5	RW	0x0	BIT_5: 0 = DISABLE 1 = ENABLE
4	RW	0x0	BIT_4: 0 = DISABLE 1 = ENABLE
3	RW	0x0	BIT_3: 0 = DISABLE 1 = ENABLE
2	RW	0x0	BIT_2: 0 = DISABLE 1 = ENABLE
1	RW	0x0	BIT_1: 0 = DISABLE 1 = ENABLE
0	RW	0x0	BIT_0: 0 = DISABLE 1 = ENABLE

8.11.14 GPIO_MSK_INT_LVL_0

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Masked Write Interrupt Activation Levels

Offset: e0h..efh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for write 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for write 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for write 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for write 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for write 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for write 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for write 1 = ENABLE
7	RW	0x0	BIT_7: 0 = LOW 1 = HIGH

Bit	R/W	Reset	Description
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

8.12 Pinmux Registers

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

This section defines the Pinmux selects, Pullup, Pulldn, and E_input programmability for each pin/ball in Tegra K1 devices. Every register has 7 bits defined for Functional Pin Select. The rest are specific to a few pads such as OD and IO_RESET.

1:0 PM --> Functional Pin Select.

3:2 PUPD --> Pullup / PullDn control.

4 Tristate --> Tristate Enable / Disable.

5 E_INPUT --> Input Receiver Enable/Disable.

7 LOCK --> Locks access to all configurable bits for pins, including itself.

--> 1 = Enable (preferably done during boot)

--> 0 = Disable (can only be done by full system reset)

8 IO_RESET --> Forces output drivers disabled (PU/PD control only). Required to guarantee pin state on *LPDDR2* pads during POR.

--> IO_RESET_N will override the enable signal of main driver. After power-up, core controls should be enabled first to drive the pad with the same value as programmed by E_PULL. Then IO_RESET_N / E_PULL can be deasserted.

--> If IO_RESET_N is deasserted before core controls come up, there could be unknown outputs or glitches.

Optional bits:

6 OD --> Open Drain Enable / Disable. Specifically for I2C pads.

9 RCV_SEL --> Select between High and Normal VIL/VIH receivers. RCVR_SEL=1: High VIL/VIH RCVR_SEL=0: Normal VIL/VIH

8.12.1 PINMUX_AUX_ULPI_DATA0_0

Offset: 0x3000 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI3	PM: 0 = SPI3 1 = HSI 2 = UARTA 3 = ULPI

8.12.2 PINMUX_AUX_ULPI_DATA1_0

Offset: 0x3004 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI3	PM: 0 = SPI3 1 = HSI 2 = UARTA 3 = ULPI

8.12.3 PINMUX_AUX_ULPI_DATA2_0

Offset: 0x3008 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI3	PM: 0 = SPI3 1 = HSI 2 = UARTA 3 = ULPI

8.12.4 PINMUX_AUX_ULPI_DATA3_0

Offset: 0x300c | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI3	PM: 0 = SPI3 1 = HSI 2 = UARTA 3 = ULPI

8.12.5 PINMUX_AUX_ULPI_DATA4_0

Offset: 0x3010 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	SPI2	PM: 0 = SPI2 1 = HSI 2 = UARTA 3 = ULPI

8.12.6 PINMUX_AUX_ULPI_DATA5_0

Offset: 0x3014 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI2	PM: 0 = SPI2 1 = HSI 2 = UARTA 3 = ULPI

8.12.7 PINMUX_AUX_ULPI_DATA6_0

Offset: 0x3018 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI2	PM: 0 = SPI2 1 = HSI 2 = UARTA 3 = ULPI

8.12.8 PINMUX_AUX_ULPI_DATA7_0

Offset: 0x301c | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI2	PM: 0 = SPI2 1 = HSI 2 = UARTA 3 = ULPI

8.12.9 PINMUX_AUX_ULPI_CLK_0

Offset: 0x3020 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI1	PM: 0 = SPI1 1 = SPI5 2 = UARTD 3 = ULPI

8.12.10 PINMUX_AUX_ULPI_DIR_0

Offset: 0x3024 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI1	PM: 0 = SPI1 1 = SPI5 2 = UARTD 3 = ULPI

8.12.11 PINMUX_AUX_ULPI_NXT_0

Offset: 0x3028 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI1	PM: 0 = SPI1 1 = SPI5 2 = UARTD 3 = ULPI

8.12.12 PINMUX_AUX_ULPI_STP_0

Offset: 0x302c | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	SPI1	PM: 0 = SPI1 1 = SPI5 2 = UARTD 3 = ULPI

8.12.13 PINMUX_AUX_DAP3_FS_0

Offset: 0x3030 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S2	PM: 0 = I2S2 1 = SPI5 2 = DISPLAYA 3 = DISPLAYB

8.12.14 PINMUX_AUX_DAP3_DIN_0

Offset: 0x3034 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S2	PM: 0 = I2S2 1 = SPI5 2 = DISPLAYA 3 = DISPLAYB

8.12.15 PINMUX_AUX_DAP3_DOUT_0

Offset: 0x3038 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S2	PM: 0 = I2S2 1 = SPI5 2 = DISPLAYA 3 = RSVD3

8.12.16 PINMUX_AUX_DAP3_SCLK_0

Offset: 0x303c | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S2	PM: 0 = I2S2 1 = SPI5 2 = RSVD2 3 = DISPLAYB

8.12.17 PINMUX_AUX_GPIO_PV0_0

Offset: 0x3040 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.18 PINMUX_AUX_GPIO_PV1_0

Offset: 0x3044 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.19 PINMUX_AUX_SDMMC1_CLK_0

Offset: 0x3048 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
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	SDMMC1	PM: 0 = SDMMC1 1 = CLK12 2 = RSVD2 3 = RSVD3

8.12.20 PINMUX_AUX_SDMMC1_CMD_0

Offset: 0x304c | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC1	PM: 0 = SDMMC1 1 = SPDIF 2 = SPI4 3 = UARTA

8.12.21 PINMUX_AUX_SDMMC1_DAT3_0

Offset: 0x3050 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC1	PM: 0 = SDMMC1 1 = SPDIF 2 = SPI4 3 = UARTA

8.12.22 PINMUX_AUX_SDMMC1_DAT2_0

Offset: 0x3054 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC1	PM: 0 = SDMMC1 1 = PWM0 2 = SPI4 3 = UARTA

8.12.23 PINMUX_AUX_SDMMC1_DAT1_0

Offset: 0x3058 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC1	PM: 0 = SDMMC1 1 = PWM1 2 = SPI4 3 = UARTA

8.12.24 PINMUX_AUX_SDMMC1_DAT0_0

Offset: 0x305c | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

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	SDMMC1	PM: 0 = SDMMC1 1 = RSVD1 2 = SPI4 3 = UARTA

8.12.25 PINMUX_AUX_CLK2_OUT_0

Offset: 0x3068 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	EXTPERIPH2	PM: 0 = EXTPERIPH2 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.26 PINMUX_AUX_CLK2_REQ_0

Offset: 0x306c | Read/Write: R/W | Reset: 0x00000020 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x100000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	DAP	PM: 0 = DAP 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.27 PINMUX_AUX_HDMI_INT_0

Offset: 0x3110 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxx0x0x110100)

Bit	Reset	Description
9	0x0	RCV_SEL: 0 = NORMAL 1 = HIGH
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.28 PINMUX_AUX_DDC_SCL_0

Offset: 0x3114 | Read/Write: R/W | Reset: 0x00000230 (0bxxxxxxxxxxxxxxxxxxxxxxx1x0x110000)

Bit	Reset	Description
9	0x1	RCV_SEL: 0 = NORMAL 1 = HIGH
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2C4	PM: 0 = I2C4 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.29 PINMUX_AUX_DDC_SDA_0

Offset: 0x3118 | Read/Write: R/W | Reset: 0x00000230 (0bxxxxxxxxxxxxxxxxxxxxxxx1x0x110000)

Bit	Reset	Description
9	0x1	RCV_SEL: 0 = NORMAL 1 = HIGH
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2C4	PM: 0 = I2C4 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.30 PINMUX_AUX_UART2_RXD_0

Offset: 0x3164 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	IRDA	PM: 0 = IRDA 1 = SPDIF 2 = UARTA 3 = SPI4

8.12.31 PINMUX_AUX_UART2_TXD_0

Offset: 0x3168 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE

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	IRDA	PM: 0 = IRDA 1 = SPDIF 2 = UARTA 3 = SPI4

8.12.32 PINMUX_AUX_UART2_RTS_N_0

Offset: 0x316c | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTA	PM: 0 = UARTA 1 = UARTB 2 = GMI 3 = SPI4

8.12.33 PINMUX_AUX_UART2_CTS_N_0

Offset: 0x3170 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	UARTA	PM: 0 = UARTA 1 = UARTB 2 = GMI 3 = SPI4

8.12.34 PINMUX_AUX_UART3_TXD_0

Offset: 0x3174 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTC	PM: 0 = UARTC 1 = RSVD1 2 = GMI 3 = SPI4

8.12.35 PINMUX_AUX_UART3_RXD_0

Offset: 0x3178 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTC	PM: 0 = UARTC 1 = RSVD1 2 = GMI 3 = SPI4

8.12.36 PINMUX_AUX_UART3_CTS_N_0

Offset: 0x317c | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTC	PM: 0 = UARTC 1 = SDMMC1 2 = DTV 3 = GMI

8.12.37 PINMUX_AUX_UART3_RTS_N_0

Offset: 0x3180 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTC	PM: 0 = UARTC 1 = PWM0 2 = DTV 3 = GMI

8.12.38 PINMUX_AUX_GPIO_PU0_0

Offset: 0x3184 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	OWR	PM: 0 = OWR 1 = UARTA 2 = GMI 3 = RSVD3

8.12.39 PINMUX_AUX_GPIO_PU1_0

Offset: 0x3188 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = UARTA 2 = GMI 3 = RSVD3

8.12.40 PINMUX_AUX_GPIO_PU2_0

Offset: 0x318c | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	RSVD0	PM: 0 = RSVD0 1 = UARTA 2 = GMI 3 = RSVD3

8.12.41 PINMUX_AUX_GPIO_PU3_0

Offset: 0x3190 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PWM0	PM: 0 = PWM0 1 = UARTA 2 = GMI 3 = DISPLAYB

8.12.42 PINMUX_AUX_GPIO_PU4_0

Offset: 0x3194 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PWM1	PM: 0 = PWM1 1 = UARTA 2 = GMI 3 = DISPLAYB

8.12.43 PINMUX_AUX_GPIO_PU5_0

Offset: 0x3198 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PWM2	PM: 0 = PWM2 1 = UARTA 2 = GMI 3 = DISPLAYB

8.12.44 PINMUX_AUX_GPIO_PU6_0

Offset: 0x319c | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PWM3	PM: 0 = PWM3 1 = UARTA 2 = RSVD2 3 = GMI

8.12.45 PINMUX_AUX_GEN1_I2C_SDA_0

Offset: 0x31a0 | Read/Write: R/W | Reset: 0x00000070 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx01110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2C1	PM: 0 = I2C1 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.46 PINMUX_AUX_GEN1_I2C_SCL_0

Offset: 0x31a4 | Read/Write: R/W | Reset: 0x00000070 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2C1	PM: 0 = I2C1 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.47 PINMUX_AUX_DAP4_FS_0

Offset: 0x31a8 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE

Bit	Reset	Description
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S3	PM: 0 = I2S3 1 = GMI 2 = DTV 3 = RSVD3

8.12.48 PINMUX_AUX_DAP4_DIN_0

Offset: 0x31ac | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S3	PM: 0 = I2S3 1 = GMI 2 = RSVD2 3 = RSVD3

8.12.49 PINMUX_AUX_DAP4_DOUT_0

Offset: 0x31b0 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S3	PM: 0 = I2S3 1 = GMI 2 = DTV 3 = RSVD3

8.12.50 PINMUX_AUX_DAP4_SCLK_0

Offset: 0x31b4 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S3	PM: 0 = I2S3 1 = GMI 2 = RSVD2 3 = RSVD3

8.12.51 PINMUX_AUX_CLK3_OUT_0

Offset: 0x31b8 | Read/Write: R/W | Reset: 0x00000020 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x100000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	EXTPERIPH3	PM: 0 = EXTPERIPH3 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.52 PINMUX_AUX_CLK3_REQ_0

Offset: 0x31bc | Read/Write: R/W | Reset: 0x00000020 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x100000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	DEV3	PM: 0 = DEV3 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.53 PINMUX_AUX_GPIO_PC7_0

Offset: 0x31c0 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = RSVD1 2 = GMI 3 = GMI_ALT

8.12.54 PINMUX_AUX_GPIO_PI5_0

Offset: 0x31c4 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	GMI	PM: 0 = SDMMC2A 1 = RSVD1 2 = GMI 3 = RSVD3

8.12.55 PINMUX_AUX_GPIO_PI7_0

Offset: 0x31c8 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = TRACE 2 = GMI 3 = DTV

8.12.56 PINMUX_AUX_GPIO_PK0_0

Offset: 0x31cc | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = SDMMC3 2 = GMI 3 = SOC

8.12.57 PINMUX_AUX_GPIO_PK1_0

Offset: 0x31d0 | Read/Write: R/W | Reset: 0x00000036 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110110)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC2A 1 = TRACE 2 = GMI 3 = RSVD3

8.12.58 PINMUX_AUX_GPIO_PJ0 _0

Offset: 0x31d4 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = RSVD1 2 = GMI 3 = USB

8.12.59 PINMUX_AUX_GPIO_PJ2_0

Offset: 0x31d8 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = RSVD1 2 = GMI 3 = SOC

8.12.60 PINMUX_AUX_GPIO_PK3_0

Offset: 0x31dc | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC2A 1 = TRACE 2 = GMI 3 = CCLA

8.12.61 PINMUX_AUX_GPIO_PK4_0

Offset: 0x31e0 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	GMI	PM: 0 = SDMMC2A 1 = RSVD1 2 = GMI 3 = GMI_ALT

8.12.62 PINMUX_AUX_GPIO_PK2_0

Offset: 0x31e4 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = RSVD1 2 = GMI 3 = RSVD3

8.12.63 PINMUX_AUX_GPIO_PI3_0

Offset: 0x31e8 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = RSVD1 2 = GMI 3 = SPI4

8.12.64 PINMUX_AUX_GPIO_PI6_0

Offset: 0x31ec | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = RSVD1 2 = GMI 3 = SDMMC2A

8.12.65 PINMUX_AUX_GPIO_PG0_0

Offset: 0x31f0 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = RSVD1 2 = GMI 3 = RSVD3

8.12.66 PINMUX_AUX_GPIO_PG1_0

Offset: 0x31f4 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = RSVD1 2 = GMI 3 = RSVD3

8.12.67 PINMUX_AUX_GPIO_PG2_0

Offset: 0x31f8 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = TRACE 2 = GMI 3 = RSVD3

8.12.68 PINMUX_AUX_GPIO_PG3_0

Offset: 0x31fc | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	GMI	PM: 0 = RSVD0 1 = TRACE 2 = GMI 3 = RSVD3

8.12.69 PINMUX_AUX_GPIO_PG4_0

Offset: 0x3200 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = TMDS 2 = GMI 3 = SPI4

8.12.70 PINMUX_AUX_GPIO_PG5_0

Offset: 0x3204 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = RSVD1 2 = GMI 3 = SPI4

8.12.71 PINMUX_AUX_GPIO_PG6_0

Offset: 0x3208 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = RSVD1 2 = GMI 3 = SPI4

8.12.72 PINMUX_AUX_GPIO_PG7_0

Offset: 0x320c | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = RSVD1 2 = GMI 3 = SPI4

8.12.73 PINMUX_AUX_GPIO_PH0_0

Offset: 0x3210 | Read/Write: R/W | Reset: 0x00000036 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110110)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = PWM0 1 = TRACE 2 = GMI 3 = DTV

8.12.74 PINMUX_AUX_GPIO_PH1_0

Offset: 0x3214 | Read/Write: R/W | Reset: 0x00000036 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110110)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = PWM1 1 = TMDS 2 = GMI 3 = DISPLAYA

8.12.75 PINMUX_AUX_GPIO_PH2_0

Offset: 0x3218 | Read/Write: R/W | Reset: 0x00000036 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110110)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	GMI	PM: 0 = PWM2 1 = TMDS 2 = GMI 3 = CLDVFS

8.12.76 PINMUX_AUX_GPIO_PH3_0

Offset: 0x321c | Read/Write: R/W | Reset: 0x00000036 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110110)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = PWM3 1 = SPI4 2 = GMI 3 = CLDVFS

8.12.77 PINMUX_AUX_GPIO_PH4_0

Offset: 0x3220 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC2A 1 = RSVD1 2 = GMI 3 = RSVD3

8.12.78 PINMUX_AUX_GPIO_PH5_0

Offset: 0x3224 | Read/Write: R/W | Reset: 0x00000036 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110110)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC2A 1 = RSVD1 2 = GMI 3 = RSVD3

8.12.79 PINMUX_AUX_GPIO_PH6_0

Offset: 0x3228 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC2A 1 = TRACE 2 = GMI 3 = DTV

8.12.80 PINMUX_AUX_GPIO_PH7_0

Offset: 0x322c | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC2A 1 = TRACE 2 = GMI 3 = DTV

8.12.81 PINMUX_AUX_GPIO_PJ7_0

Offset: 0x3230 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = UARTD 1 = RSVD1 2 = GMI 3 = GMI_ALT

8.12.82 PINMUX_AUX_GPIO_PB0_0

Offset: 0x3234 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

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

Bit	Reset	Description
1:0	GMI	PM: 0 = UARTD 1 = RSVD1 2 = GMI 3 = RSVD3

8.12.83 PINMUX_AUX_GPIO_PB1_0

Offset: 0x3238 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = UARTD 1 = RSVD1 2 = GMI 3 = RSVD3

8.12.84 PINMUX_AUX_GPIO_PK7_0

Offset: 0x323c | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = UARTD 1 = RSVD1 2 = GMI 3 = RSVD3

8.12.85 PINMUX_AUX_GPIO_PI0_0

Offset: 0x3240 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = RSVD1 2 = GMI 3 = RSVD3

8.12.86 PINMUX_AUX_GPIO_PI1_0

Offset: 0x3244 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = RSVD1 2 = GMI 3 = RSVD3

8.12.87 PINMUX_AUX_GPIO_PI2_0

Offset: 0x3248 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC2A 1 = TRACE 2 = GMI 3 = RSVD3

8.12.88 PINMUX_AUX_GPIO_PI4_0

Offset: 0x324c | Read/Write: R/W | Reset: 0x00000036 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110110)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SPI4 1 = TRACE 2 = GMI 3 = DISPLAYA

8.12.89 PINMUX_AUX_GEN2_I2C_SCL_0

Offset: 0x3250 | Read/Write: R/W | Reset: 0x00000070 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	I2C2	PM: 0 = I2C2 1 = RSVD1 2 = GMI 3 = RSVD3

8.12.90 PINMUX_AUX_GEN2_I2C_SDA_0

Offset: 0x3254 | Read/Write: R/W | Reset: 0x00000070 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2C2	PM: 0 = I2C2 1 = RSVD1 2 = GMI 3 = RSVD3

8.12.91 PINMUX_AUX_SDMMC4_CLK_0

Offset: 0x3258 | Read/Write: R/W | Reset: 0x00000136 (0bxxxxxxxxxxxxxxxxxxxxxxxx10x110110)

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC4 1 = RSVD1 2 = GMI 3 = RSVD3

8.12.92 PINMUX_AUX_SDMMC4_CMD_0

Offset: 0x325c | Read/Write: R/W | Reset: 0x0000013a (0bxxxxxxxxxxxxxxxxxxxxxxxx10x111010)

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC4 1 = RSVD1 2 = GMI 3 = RSVD3

8.12.93 PINMUX_AUX_SDMMC4_DAT0_0

Offset: 0x3260 | Read/Write: R/W | Reset: 0x0000013a (0bxxxxxxxxxxxxxxxxxxxxxxxx10x111010)

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC4 1 = SPI3 2 = GMI 3 = RSVD3

8.12.94 PINMUX_AUX_SDMMC4_DAT1_0

Offset: 0x3264 | Read/Write: R/W | Reset: 0x0000013a (0bxxxxxxxxxxxxxxxxxxxxxxxx10x111010)

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC4 1 = SPI3 2 = GMI 3 = RSVD3

8.12.95 PINMUX_AUX_SDMMC4_DAT2_0

Offset: 0x3268 | Read/Write: R/W | Reset: 0x0000013a (0bxxxxxxxxxxxxxxxxxxxxxxxx10x111010)

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC4 1 = SPI3 2 = GMI 3 = RSVD3

8.12.96 PINMUX_AUX_SDMMC4_DAT3_0

Offset: 0x326c | Read/Write: R/W | Reset: 0x0000013a (0bxxxxxxxxxxxxxxxxxxxxxxxx10x111010)

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC4 1 = SPI3 2 = GMI 3 = RSVD3

8.12.97 PINMUX_AUX_SDMMC4_DAT4_0

Offset: 0x3270 | Read/Write: R/W | Reset: 0x0000013a (0bxxxxxxxxxxxxxxxxxxxxxxxx10x111010)

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC4 1 = SPI3 2 = GMI 3 = RSVD3

8.12.98 PINMUX_AUX_SDMMC4_DAT5_0

Offset: 0x3274 | Read/Write: R/W | Reset: 0x0000013a (0bxxxxxxxxxxxxxxxxxxxxxxxx10x111010)

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD2	PM: 0 = SDMMC4 1 = SPI3 2 = RSVD2 3 = RSVD3

8.12.99 PINMUX_AUX_SDMMC4_DAT6_0

Offset: 0x3278 | Read/Write: R/W | Reset: 0x0000013a (0bxxxxxxxxxxxxxxxxxxxxxxxx10x111010)

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC4 1 = SPI3 2 = GMI 3 = RSVD3

8.12.100 PINMUX_AUX_SDMMC4_DAT7_0

Offset: 0x327c | Read/Write: R/W | Reset: 0x0000013a (0bxxxxxxxxxxxxxxxxxxxxxxxx10x111010)

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC4 1 = RSVD1 2 = GMI 3 = RSVD3

8.12.101 PINMUX_AUX_CAM_MCLK_0

Offset: 0x3284 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

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 = SDMMC2B

8.12.102 PINMUX_AUX_GPIO_PCC1_0

Offset: 0x3288 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

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	I2S4	PM: 0 = I2S4 1 = RSVD1 2 = RSVD2 3 = SDMMC2B

8.12.103 PINMUX_AUX_GPIO_PBB0_0

Offset: 0x328c | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	VGP6	PM: 0 = VGP6 1 = VIMCLK2 2 = SDMMC2B 3 = VIMCLK2_ALT

8.12.104 PINMUX_AUX_CAM_I2C_SCL_0

Offset: 0x3290 | Read/Write: R/W | Reset: 0x00000070 (0bxxxxxxxxxxxxxxxxxxxxxxxx01110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE

Bit	Reset	Description
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	VGP1	PM: 0 = VGP1 1 = I2C3 2 = RSVD2 3 = SDMMC2B

8.12.105 PINMUX_AUX_CAM_I2C_SDA_0

Offset: 0x3294 | Read/Write: R/W | Reset: 0x00000070 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	VGP2	PM: 0 = VGP2 1 = I2C3 2 = RSVD2 3 = SDMMC2B

8.12.106 PINMUX_AUX_GPIO_PBB3_0

Offset: 0x3298 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	VGP3	PM: 0 = VGP3 1 = DISPLAYA 2 = DISPLAYB 3 = SDMMC2B

8.12.107 PINMUX_AUX_GPIO_PBB4_0

Offset: 0x329c | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	VGP4	PM: 0 = VGP4 1 = DISPLAYA 2 = DISPLAYB 3 = SDMMC2B

8.12.108 PINMUX_AUX_GPIO_PBB5_0

Offset: 0x32a0 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	VGP5	PM: 0 = VGP5 1 = DISPLAYA 2 = RSVD2 3 = SDMMC2B

8.12.109 PINMUX_AUX_GPIO_PBB6_0

Offset: 0x32a4 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S4	PM: 0 = I2S4 1 = RSVD1 2 = DISPLAYB 3 = SDMMC2B

8.12.110 PINMUX_AUX_GPIO_PBB7_0

Offset: 0x32a8 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S4	PM: 0 = I2S4 1 = RSVD1 2 = RSVD2 3 = SDMMC2B

8.12.111 PINMUX_AUX_GPIO_PCC2_0

Offset: 0x32ac | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S4	PM: 0 = I2S4 1 = RSVD1 2 = SDMMC3 3 = SDMMC2B

8.12.112 PINMUX_AUX_JTAG_RTCK_0

Offset: 0x32b0 | Read/Write: R/W | Reset: 0x00000028 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x101000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RTCK	PM: 0 = RTCK 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.113 PINMUX_AUX_PWR_I2C_SCL_0

Offset: 0x32b4 | Read/Write: R/W | Reset: 0x00000070 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	I2CPWR	PM: 0 = I2CPWR 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.114 PINMUX_AUX_PWR_I2C_SDA_0

Offset: 0x32b8 | Read/Write: R/W | Reset: 0x00000070 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2CPWR	PM: 0 = I2CPWR 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.115 PINMUX_AUX_KB_ROW0_0

Offset: 0x32bc | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.116 PINMUX_AUX_KB_ROW1_0

Offset: 0x32c0 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.117 PINMUX_AUX_KB_ROW2_0

Offset: 0x32c4 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.118 PINMUX_AUX_KB_ROW3_0

Offset: 0x32c8 | Read/Write: R/W | Reset: 0x00000022 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x100010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SYS	PM: 0 = KBC 1 = DISPLAYA 2 = SYS 3 = DISPLAYB

8.12.119 PINMUX_AUX_KB_ROW4_0

Offset: 0x32cc | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = DISPLAYA 2 = RSVD2 3 = DISPLAYB

8.12.120 PINMUX_AUX_KB_ROW5_0

Offset: 0x32d0 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	KBC	PM: 0 = KBC 1 = DISPLAYA 2 = RSVD2 3 = DISPLAYB

8.12.121 PINMUX_AUX_KB_ROW6_0

Offset: 0x32d4 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = DISPLAYA 2 = DISPLAYA_ALT 3 = DISPLAYB

8.12.122 PINMUX_AUX_KB_ROW7_0

Offset: 0x32d8 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = CLDVFS 3 = UARTA

8.12.123 PINMUX_AUX_KB_ROW8_0

Offset: 0x32dc | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = CLDVFS 3 = UARTA

8.12.124 PINMUX_AUX_KB_ROW9_0

Offset: 0x32e0 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = RSVD2 3 = UARTA

8.12.125 PINMUX_AUX_KB_ROW10_0

Offset: 0x32e4 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = RSVD2 3 = UARTA

8.12.126 PINMUX_AUX_KB_ROW11_0

Offset: 0x32e8 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = RSVD2 3 = IRDA

8.12.127 PINMUX_AUX_KB_ROW12_0

Offset: 0x32ec | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = RSVD2

Bit	Reset	Description
		3 = IRDA

8.12.128 PINMUX_AUX_KB_ROW13_0

Offset: 0x32f0 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = SPI2 3 = RSVD3

8.12.129 PINMUX_AUX_KB_ROW14_0

Offset: 0x32f4 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = SPI2 3 = RSVD3

8.12.130 PINMUX_AUX_KB_ROW15_0

Offset: 0x32f8 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = SOC 2 = RSVD2 3 = RSVD3

8.12.131 PINMUX_AUX_KB_COL0_0

Offset: 0x32fc | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = SPI2 3 = RSVD3

8.12.132 PINMUX_AUX_KB_COL1_0

Offset: 0x3300 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = SPI2 3 = RSVD3

8.12.133 PINMUX_AUX_KB_COL2_0

Offset: 0x3304 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = SPI2 3 = RSVD3

8.12.134 PINMUX_AUX_KB_COL3_0

Offset: 0x3308 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = DISPLAYA 2 = PWM2 3 = UARTA

8.12.135 PINMUX_AUX_KB_COL4_0

Offset: 0x330c | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = OWR 2 = SDMMC3 3 = UARTA

8.12.136 PINMUX_AUX_KB_COL5_0

Offset: 0x3310 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = SDMMC3 3 = RSVD3

8.12.137 PINMUX_AUX_KB_COL6_0

Offset: 0x3314 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = SPI2 3 = UARTD

8.12.138 PINMUX_AUX_KB_COL7_0

Offset: 0x3318 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = SPI2 3 = UARTD

8.12.139 PINMUX_AUX_CLK_32K_OUT_0

Offset: 0x331c | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	BLINK	PM: 0 = BLINK 1 = SOC 2 = RSVD2 3 = RSVD3

8.12.140 PINMUX_AUX_CORE_PWR_REQ_0

Offset: 0x3324 | Read/Write: R/W | Reset: 0x00000020 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x100000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PWRON	PM: 0 = PWRON 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.141 PINMUX_AUX_CPU_PWR_REQ_0

Offset: 0x3328 | Read/Write: R/W | Reset: 0x00000020 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x100000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	CPU	PM: 0 = CPU 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.142 PINMUX_AUX_PWR_INT_N_0

Offset: 0x332c | Read/Write: R/W | Reset: 0x00000020 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x100000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PMI	PM: 0 = PMI 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.143 PINMUX_AUX_CLK_32K_IN_0

Offset: 0x3330 | Read/Write: R/W | Reset: 0x00000020 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x100000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	CLK	PM: 0 = CLK 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.144 PINMUX_AUX_OWR_0

Offset: 0x3334 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxx0x0x110000)

Bit	Reset	Description
9	0x0	RCV_SEL: 0 = NORMAL 1 = HIGH
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL

Bit	Reset	Description
		1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	OWR	PM: 0 = OWR 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.145 PINMUX_AUX_DAP1_FS_0

Offset: 0x3338 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S0	PM: 0 = I2S0 1 = HDA 2 = GMI 3 = RSVD3

8.12.146 PINMUX_AUX_DAP1_DIN_0

Offset: 0x333c | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S0	PM: 0 = I2S0 1 = HDA 2 = GMI 3 = RSVD3

8.12.147 PINMUX_AUX_DAP1_DOUT_0

Offset: 0x3340 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S0	PM: 0 = I2S0 1 = HDA 2 = GMI 3 = SATA

8.12.148 PINMUX_AUX_DAP1_SCLK_0

Offset: 0x3344 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S0	PM: 0 = I2S0 1 = HDA 2 = GMI 3 = RSVD3

8.12.149 PINMUX_AUX_DAP_MCLK1_REQ_0

Offset: 0x3348 | Read/Write: R/W | Reset: 0x00000024 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x100100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	DAP	PM: 0 = DAP 1 = DAP1 2 = SATA 3 = RSVD3

8.12.150 PINMUX_AUX_DAP_MCLK1_0

Offset: 0x334c | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	EXTPERIPH1	PM: 0 = EXTPERIPH1 1 = DAP2 2 = RSVD2 3 = RSVD3

8.12.151 PINMUX_AUX_SPDIF_IN_0

Offset: 0x3350 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPDIF	PM: 0 = SPDIF 1 = RSVD1

Bit	Reset	Description
		2 = RSVD2 3 = I2C3

8.12.152 PINMUX_AUX_SPDIF_OUT_0

Offset: 0x3354 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPDIF	PM: 0 = SPDIF 1 = RSVD1 2 = RSVD2 3 = I2C3

8.12.153 PINMUX_AUX_DAP2_FS_0

Offset: 0x3358 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S1	PM: 0 = I2S1 1 = HDA 2 = GMI 3 = RSVD3

8.12.154 PINMUX_AUX_DAP2_DIN_0

Offset: 0x335c | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S1	PM: 0 = I2S1 1 = HDA 2 = GMI 3 = RSVD3

8.12.155 PINMUX_AUX_DAP2_DOUT_0

Offset: 0x3360 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S1	PM: 0 = I2S1 1 = HDA 2 = GMI 3 = RSVD3

8.12.156 PINMUX_AUX_DAP2_SCLK_0

Offset: 0x3364 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S1	PM: 0 = I2S1 1 = HDA 2 = GMI 3 = RSVD3

8.12.157 PINMUX_AUX_DVFS_PWM_0

Offset: 0x3368 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI6	PM: 0 = SPI6 1 = CLDVFS 2 = GMI 3 = RSVD3

8.12.158 PINMUX_AUX_GPIO_X1_AUD_0

Offset: 0x336c | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI6	PM: 0 = SPI6 1 = RSVD1

Bit	Reset	Description
		2 = GMI 3 = RSVD3

8.12.159 PINMUX_AUX_GPIO_X3_AUD_0

Offset: 0x3370 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI6	PM: 0 = SPI6 1 = SPI1 2 = GMI 3 = RSVD3

8.12.160 PINMUX_AUX_DVFS_CLK_0

Offset: 0x3374 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI6	PM: 0 = SPI6 1 = CLDVFS 2 = GMI 3 = RSVD3

8.12.161 PINMUX_AUX_GPIO_X4_AUD_0

Offset: 0x3378 | Read/Write: R/W | Reset: 0x00000035 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110101)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
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 = GMI 1 = SPI1 2 = SPI2 3 = DAP2

8.12.162 PINMUX_AUX_GPIO_X5_AUD_0

Offset: 0x337c | Read/Write: R/W | Reset: 0x00000039 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111001)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI1	PM: 0 = GMI 1 = SPI1 2 = SPI2 3 = RSVD3

8.12.163 PINMUX_AUX_GPIO_X6_AUD_0

Offset: 0x3380 | Read/Write: R/W | Reset: 0x00000039 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111001)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

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	SPI1	PM: 0 = SPI6 1 = SPI1 2 = SPI2 3 = GMI

8.12.164 PINMUX_AUX_GPIO_X7_AUD_0

Offset: 0x3384 | Read/Write: R/W | Reset: 0x00000035 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110101)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI1	PM: 0 = RSVD0 1 = SPI1 2 = SPI2 3 = RSVD3

8.12.165 PINMUX_AUX_SDMMC3_CLK_0

Offset: 0x3390 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	SDMMC3	PM: 0 = SDMMC3 1 = RSVD1 2 = RSVD2 3 = SPI3

8.12.166 PINMUX_AUX_SDMMC3_CMD_0

Offset: 0x3394 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = PWM3 2 = UARTA 3 = SPI3

8.12.167 PINMUX_AUX_SDMMC3_DAT0_0

Offset: 0x3398 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = RSVD1 2 = RSVD2 3 = SPI3

8.12.168 PINMUX_AUX_SDMMC3_DAT1_0

Offset: 0x339c | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = PWM2 2 = UARTA 3 = SPI3

8.12.169 PINMUX_AUX_SDMMC3_DAT2_0

Offset: 0x33a0 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = PWM1 2 = DISPLAYA 3 = SPI3

8.12.170 PINMUX_AUX_SDMMC3_DAT3_0

Offset: 0x33a4 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = PWM0 2 = DISPLAYB 3 = SPI3

8.12.171 PINMUX_AUX_PEX_L0_RST_N_0

Offset: 0x33bc | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PE0	PM: 0 = PE0 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.172 PINMUX_AUX_PEX_L0_CLKREQ_N_0

Offset: 0x33c0 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PE0	PM: 0 = PE0 1 = RSVD1 2 = RSVD2

Bit	Reset	Description
		3 = RSVD3

8.12.173 PINMUX_AUX_PEX_WAKE_N_0

Offset: 0x33c4 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PE	PM: 0 = PE 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.174 PINMUX_AUX_PEX_L1_RST_N_0

Offset: 0x33cc | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PE1	PM: 0 = PE1 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.175 PINMUX_AUX_PEX_L1_CLKREQ_N_0

Offset: 0x33d0 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE

Bit	Reset	Description
		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	PE1	PM: 0 = PE1 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.176 PINMUX_AUX_HDMI_CEC_0

Offset: 0x33e0 | Read/Write: R/W | Reset: 0x00000070 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	CEC	PM: 0 = CEC 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.177 PINMUX_AUX_SDMMC1_WP_N_0

Offset: 0x33e4 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE

Bit	Reset	Description
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC1	PM: 0 = SDMMC1 1 = CLK12 2 = SPI4 3 = UARTA

8.12.178 PINMUX_AUX_SDMMC3_CD_N_0

Offset: 0x33e8 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = OWR 2 = RSVD2 3 = RSVD3

8.12.179 PINMUX_AUX_GPIO_W2_AUD_0

Offset: 0x33ec | Read/Write: R/W | Reset: 0x00000039 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111001)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD1	PM: 0 = SPI6 1 = RSVD1 2 = SPI2 3 = I2C1

8.12.180 PINMUX_AUX_GPIO_W3_AUD_0

Offset: 0x33f0 | Read/Write: R/W | Reset: 0x00000039 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111001)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI1	PM: 0 = SPI6 1 = SPI1 2 = SPI2 3 = I2C1

8.12.181 PINMUX_AUX_USB_VBUS_EN0_0

Offset: 0x33f4 | Read/Write: R/W | Reset: 0x00000060 (0bxxxxxxxxxxxxxxxxxxxxxxxx01100000)

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	USB	PM: 0 = USB 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.182 PINMUX_AUX_USB_VBUS_EN1_0

Offset: 0x33f8 | Read/Write: R/W | Reset: 0x00000060 (0bxxxxxxxxxxxxxxxxxxxxxxxx01100000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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	USB	PM: 0 = USB 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.183 PINMUX_AUX_SDMMC3_CLK_LB_IN_0

Offset: 0x33fc | Read/Write: R/W | Reset: 0x00000024 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x100100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.184 PINMUX_AUX_SDMMC3_CLK_LB_OUT_0

Offset: 0x3400 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL

Bit	Reset	Description
		1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.185 PINMUX_AUX_ GMI_CLK_LB_0

Offset: 0x3404 | Read/Write: R/W | Reset: 0x00000036 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110110)

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 = SDMMC2A 1 = RSVD1 2 = GMI 3 = RSVD3

8.12.186 PINMUX_AUX_RESET_OUT_N_0

Offset: 0x3408 | Read/Write: R/W | Reset: 0x00000020 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x100000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RESET_OUT_N

8.12.187 PINMUX_AUX_KB_ROW16_0

Offset: 0x340c | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = RSVD2 3 = UARTC

8.12.188 PINMUX_AUX_KB_ROW17_0

Offset: 0x3410 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = RSVD2 3 = UARTC

8.12.189 PINMUX_AUX_USB_VBUS_EN2_0

Offset: 0x3414 | Read/Write: R/W | Reset: 0x00000060 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01100000)

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

Bit	Reset	Description
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	USB	PM: 0 = USB 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.190 PINMUX_AUX_GPIO_PFF2_0

Offset: 0x3418 | Read/Write: R/W | Reset: 0x00000070 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SATA	PM: 0 = SATA 1 = RSVD1 2 = RSVD2 3 = RSVD3

8.12.191 PINMUX_AUX_DP_HPD_0

Offset: 0x3430 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	DP	PM: 0 = DP 1 = RSVD1 2 = RSVD2 3 = RSVD3

[THIS PAGE INTENTIONALLY LEFT BLANK]

9.0 POWER MANAGEMENT CONTROLLER

9.1 Register Definitions for the PMC

To speed up operation, the PMC register file operates in the local peripheral interface bus domain (APB) rather than in the 32 kHz clock domain used for PMC processing.

Registers in the PMC control entering/exiting Deep Sleep/Suspend mode, power detect, and I/O power functions. They also control CORE_PWR_REQ, SYS_CLK_REQ, and LED_BLINK pins.

The following registers should be programmed/read for different power events in order for PMC to function properly.

IMPORTANT:	**NO_IO_POWER REGISTER (APBDEV_PMC_NO_IOPOWER_0)** Before 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.

BEFORE/ON ENTERING DEEP SLEEP MODE

- PMC Control Register (bits PWRREQ_POLARITY, PWRREQ_OE, SYSCLK_POLARITY,SYSCLK_OE, LATCHWAKE_EN)
- WAKE_MASK
- WAKE_LVL
- DPD_PADS_ORIDE (for required overrides)
- PWRGOOD_TIMER
- DPD_SAMPLE
- DPD_ENABLE

AFTER WAKE-UP FROM DEEP SLEEP

- WAKE_STATUS
- SW_WAKE_STATUS
- DPD_ENABLE
- PMC Control Register (bit LATCHWAKE_EN)

BEFORE/ON POWERING DOWN PARTITIONS

- PWRGATE_STATUS
- PWRGATE_TIMER_OFF (for power down)
- PWRGATE_TIMER_ON (for power up). This has been deprecated.
- PWRGATE_TOGGLE
- REMOVE_CLAMPING_CMD (for power up)

FOR POWER DETECT SEQUENCE

- PWR_DET
- PWR_DET_LATCH

FOR BLINK

- BLINK_TIMER
- PMC Control Register (bit BLINK_EN)
- DPD_PADS_ORIDE (for blink field)

All other operations require single register access.

9.2 PMC Registers

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

9.2.1 APBDEV_PMC_CNTRL_0

This register controls the clock to the RTC, and the reset to the CAR and RTC. It also manages polarity and output enable of the sys_clk_req and core_pwr_req pins. At reset, both are disabled since the PMIC properties are unknown.

Auxiliary functions include enabling of blinking functions, side-effect option, and software wake-up latching.

The LP0 entry is to be triggered by the PMC side-effect option. Software needs to configure the PMC for the side-effect option before triggering the power off of last CPU. If the side-effect option is set, then the PMC would initiate LP0 in the following cases:

- A power-off request for the CPU rail will lead to CPU rail power-off followed by LP0 entry.
- A power-gating request for cluster1 non-CPU (i.e., C1NC) will lead to power-gating of the C1NC followed by LP0 entry

PMC Control Register

Offset: 0x0 │ Read/Write: R/W │ Reset: 0x00010a00 (0bxxxxxxxxxx00000100001010000000000)

Bit	Reset	Description
21:20	0x0	CPUPWRGOOD_SEL: CPUPWRGOOD pin select. There are 4 potential source pins (soc_therm_oc{1,2,3,4}) used as PWRGOOD. This field needs to be programmed (at boot) based on which pin is used as PWRGOOD. 0 = SOC_THERM_OC1 (default) 1 = SOC_THERM_OC2 2 = SOC_THERM_OC3 3 = SOC_THERM_OC4
19	0x0	CPUPWRGOOD_EN:CPU_PWR_GOOD (From PMIC to PMC) signal is enabled. 0 = DISABLE 1 = ENABLE
18	0x0	FUSE_OVERRIDE: Fuse override 0 = DISABLE 1 = ENABLE
17	0x0	INTR_POLARITY: Inverts INTR polarity 0 = DISABLE 1 = ENABLE
16	0x1	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

Bit	Reset	Description
14	0x0	SIDE_EFFECT_LP0: When set, causes the side effect of entering LP0 after powering down the CPU 0 = DISABLE 1 = ENABLE
13	0x0	AOINIT: AO initialized purely software diagnostic and interpretation 0 = NOTDONE 1 = DONE
12	0x0	PWRGATE_DIS: Disable power gating - global override, will override function of PWRGATE_TOGGLE register. All partitions will stay enabled. **Note**: Only write to this bit when CRAIL is on. 0 = DISABLE 1 = ENABLE
11	0x1	SYSCLK_OE: Enables output of system enable clock - works only if the SYS_CLK field in DPD_PADS_ORIDE is set to 1. 0 = DISABLE 1 = ENABLE
10	0x0	SYSCLK_POLARITY: Inverts SYS_CLK_REQ enable polarity 0 = NORMAL 1 = INVERT
9	0x1	PWRREQ_OE: CORE_PWR_REQ output enable. Resets to tristate 0 = DISABLE 1 = ENABLE
8	0x0	PWRREQ_POLARITY: Inverts CORE_PWR_REQ polarity 0 = NORMAL 1 = INVERT
7	0x0	BLINK_EN: Enables blinking counter and blink output works only if the BLINK field in DPD_PADS_ORIDE is set to 1. 0 = DISABLE 1 = ENABLE
6	0x0	GLITCHDET_DIS: Reserved
5	0x0	LATCHWAKE_EN: Enables latching wakeup events - stops latching on transition from 1 to 0 (sequence - set to 1,set to 0) 0 = DISABLE 1 = ENABLE
4	0x0	MAIN_RST: Resets everything but scratch 0 and reset status. 0 = DISABLE 1 = ENABLE
3	0x0	KBC_RST: Reserved
2	0x0	RTC_RST: Software reset to RTC. 0 = DISABLE 1 = ENABLE
1	0x0	RTC_CLK_DIS: Disable 32 KHz clock to RTC. 0 = DISABLE 1 = ENABLE
0	0x0	KBC_CLK_DIS: Reserved

9.2.2 APBDEV_PMC_SEC_DISABLE_0

On separate reset (same as CAR), this register disables access (read/write to secure scratch registers). Once writes are disabled, secure registers cannot be written until power on reset or reset when exiting Deep Sleep mode.

Once reads are disabled, reads from scratch registers will return 0 until power on reset or reset when exiting Deep Sleep mode.

Single register can be disabled for reads/writes; bits 0 and 1 have an overwrite function.

Secure Register Disable

Offset: 0x4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
21	0x0	TSC_NS_WRITE: Non-secure (i.e., non-TZ) write-disable for the PMC_TSC_MULT register and the PMC_DPD_ENABLE[TSC_MULT_EN] register-bit. 0: Non-secure or secure can write. 1: Only secure can write. This bit is write-once only. Once it is set to '1', it remains '1' until reset by LP0 or system reset. 0 = OFF 1 = ON
20	0x0	AMAP_WRITE: Disable writes to the PMC_GLB_AMAP_CFG register 0 = OFF 1 = ON
19	0x0	READ7: Disable reads from secure register 7 0 = OFF 1 = ON
18	0x0	WRITE7: Disable writes to secure register 7 0 = OFF 1 = ON
17	0x0	READ6: Disable reads from secure register 6 0 = OFF 1 = ON
16	0x0	WRITE6: Disable writes to secure register 6 0 = OFF 1 = ON
15	0x0	READ5: Disable reads from secure register 5 0 = OFF 1 = ON
14	0x0	WRITE5: Disable writes to secure register 5 0 = OFF 1 = ON
13	0x0	READ4: Disable reads from secure register 4 0 = OFF 1 = ON
12	0x0	WRITE4: Disable writes to secure register 4 0 = OFF 1 = ON
11	0x0	READ3: Disable reads from secure register 3 0 = OFF 1 = ON

Bit	Reset	Description
10	0x0	WRITE3: Disable writes to secure register 3 0 = OFF 1 = ON
9	0x0	READ2: Disable reads from secure register 2 0 = OFF 1 = ON
8	0x0	WRITE2: Disable writes to secure register 2 0 = OFF 1 = ON
7	0x0	READ1: Disable reads from secure register 1 0 = OFF 1 = ON
6	0x0	WRITE1: Disable writes to secure register 1 0 = OFF 1 = ON
5	0x0	READ0: Disable reads from secure register 0 0 = OFF 1 = ON
4	0x0	WRITE0: Disable writes to secure register 0 0 = OFF 1 = ON
3:2	0x0	NOT_USED: legacy - redundant - not used
1	0x0	READ: Disable reads from all secure registers 0 = OFF 1 = ON
0	0x0	WRITE: Disable writes to all secure registers 0 = OFF 1 = ON

9.2.3 APBDEV_PMC_PMC_SWRST_0

Reset only by the POR cell; sets itself back to inactive after 2 clock cycles. Only used for emergency debugging purposes; not to be used in any functional mode.

Note: Defunct. Kept only for binary code compatibility only. Will not do anything

Register Write which Resets PMC Only

Offset: 0x8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	RST: Software reset to the PMC only 0 = DISABLE 1 = ENABLE

9.2.4 APBDEV_PMC_WAKE_MASK_0

The APBDEV_PMC_WAKE registers handle wake events whose number is less than or equal to 32. For wake events whose number is greater than 32, refer to the APBDEV_PMC_WAKE2 registers below.

Current pin assignments wake_mask, wake_status, sw_wake_status, and wake levels:

- WAKE_STATUS[0] = ulpi_data4 (wake event 0)
- WAKE_STATUS[1] = gp3_pv[1] (wake event 1)
- WAKE_STATUS[3] = sdmmc3_dat1 (wake event 3)
- WAKE_STATUS[4] = hdmi_int (wake event 4)
- WAKE_STATUS[6] = gp3_pu[5] (wake event 6)
- WAKE_STATUS[7] = gp3_pu[6] (wake event 7)
- WAKE_STATUS[8] = gpio_pc[7] (wake event 8)
- WAKE_STATUS[9] = kb_row10 (wake event 9)
- WAKE_STATUS[10] = sdmmc4_dat1 (wake event 10)
- WAKE_STATUS[11] = gpio_w3_aud (wake event 11)
- WAKE_STATUS[12] = gpio_w2_aud (wake event 12)
- WAKE_STATUS[13] = sdmmc1_dat1 (wake event 13)
- WAKE_STATUS[14] = pe_wake_n (wake event 14)
- WAKE_STATUS[15] = gpio_pj[2] (wake event 15)
- WAKE_STATUS[16] = rtc_irq (wake event 16)
- WAKE_STATUS[17] = kbc_interrupt (wake event 17)
- WAKE_STATUS[18] = pwr_int (wake event 18)
- WAKE_STATUS[19] = usb1_vbus (wake event 19)
- WAKE_STATUS[20] = usb_vbus_wakeup[1] (wake event 20)
- WAKE_STATUS[23] = gpio_pi[5] (wake event 23)
- WAKE_STATUS[24] = gp3_pv[0] (wake event 24)
- WAKE_STATUS[25] = kb_row12 (wake event 25)
- WAKE_STATUS[26] = kb_row13 (wake event 26)
- WAKE_STATUS[27] = kb_row8 (wake event 27)
- WAKE_STATUS[28] = kb_row14 (wake event 28)
- WAKE_STATUS[29] = kb_row15 (wake event 29)
- WAKE_STATUS[30] = dap1_dout (wake event 30)

This register masks which event can cause a wake-up from Deep Sleep mode. It has to be set up before entering Deep Sleep mode. It works in conjunction with the WAKE_LVL register.

Only enabled events at the proper wake_lvl will cause exit from Deep Sleep mode.

PMC Wake-up Event Mask

Offset: 0xc │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	RESET_N: External reset wake enable 0 = DISABLE 1 = ENABLE
30:23	0x0	EVENT_RES: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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

9.2.5 APBDEV_PMC_WAKE_LVL_0

This register sets the active level for the wake event. It will cause an exit from Deep Sleep mode if the input signal level matches the level set in this register and WAKE_MASK is set for the event to 1.

PMC Wake Level

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x7f9fffff (0b01111111100111111111111111111111)

Bit	Reset	Description
31	0x0	RESET_N: External reset wake level (low active) 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
30:23	0xff	EVENT_RES: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
22:19	0x3	USB_EVENT: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
18	0x1	PWR_INT_N: Power interrupt - permanently tied to bit 18 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
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

9.2.6 APBDEV_PMC_WAKE_STATUS_0

This register stores the status of the wake events. The event will be set if the level matches and is not masked.

A write will reset the set wake event.

PMC Wake Status

Offset: 0x14 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	RESET_N: external reset 0 = NOT_SET 1 = SET
30:23	0x0	EVENT_RES: 0 = NOT_SET 1 = SET
22:19	0x0	USB_EVENT: USB wake events 0 = NOT_SET 1 = SET
18	0x0	PWR_INT_N: power interrupt 0 = NOT_SET 1 = SET
17	0x0	KBC: KBC wake 0 = NOT_SET 1 = SET
16	0x0	RTC: RTC wake 0 = NOT_SET 1 = SET
15:0	0x0	EVENT: pin 0-15 wake 0 = NOT_SET 1 = SET

9.2.7 APBDEV_PMC_SW_WAKE_STATUS_0

This register stores status of the wake events. Latching of the events in software wake status is enabled by the PMC_CNTRL register bit LATCHWAKE_EN.

Latching will stop at a 1-to-0 transition on this bit. An event will be set if the level matches. Masking does not affect this register. A write will reset the set wake events.

PMC Software Wake Status

Offset: 0x18 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	RESET_N: External reset 0 = NOT_SET 1 = SET

Bit	Reset	Description
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

9.2.8 APBDEV_PMC_DPD_PADS_ORIDE_0

This register enables overriding values from pinmux with values driven by the KBC (keyboard) or the PMC (sys_clk_req, blink).

If a bit is set to 1, the associated I/O will drive the direct value from the KBC or PMC, not the pinmux value. During Deep Sleep mode, the pads with the bit set to 1 will not be in Deep Power Down mode. It will not drive data from mini pad macros stored during sample cycle, but direct data from the KBC or PMC.

Note: This register has to be set before entering Deep Sleep mode.

DPD Pads Override

Offset: 0x1c │ Read/Write: R/W │ Reset: 0x00200000 (0b00000000001000000000000000000000)

Bit	Reset	Description
31	0x0	KBC_ROW16: Override DPD idle state with row 16 output 0 = DISABLE 1 = ENABLE
30	0x0	KBC_ROW15: Override DPD idle state with row 15 output 0 = DISABLE 1 = ENABLE
29	0x0	KBC_ROW14: Override DPD idle state with row 14 output 0 = DISABLE 1 = ENABLE
28	0x0	KBC_ROW13: Override DPD idle state with row 13 output 0 = DISABLE 1 = ENABLE
27	0x0	KBC_ROW12: Override DPD idle state with row 12 output 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
26	0x0	KBC_ROW11: Override DPD idle state with row 11 output 0 = DISABLE 1 = ENABLE
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 of the gp3_pq[7] pad 0 = DISABLE 1 = ENABLE
17	0x0	KBC_ROW4: Override DPD idle state of the gp3_pq[6] pad 0 = DISABLE 1 = ENABLE
16	0x0	KBC_ROW3: Override DPD idle state of the gp3_pq[5] pad 0 = DISABLE 1 = ENABLE
15	0x0	KBC_ROW2: Override DPD idle state of the gp3_pq[4] pad 0 = DISABLE 1 = ENABLE
14	0x0	KBC_ROW1: Override DPD idle state of the gp3_pq[3] pad 0 = DISABLE 1 = ENABLE
13	0x0	KBC_ROW0: Override DPD idle state of the gp3_pq[2] pad 0 = DISABLE 1 = ENABLE
12	0x0	KBC_COL12: Override DPD idle state with row 17 output 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	0x0	KBC_COL11: Override DPD idle state of the gp3_pq[0] pad 0 = DISABLE 1 = ENABLE
10	0x0	KBC_COL10: Override DPD idle state of the gp3_ps[2] pad 0 = DISABLE 1 = ENABLE
9	0x0	KBC_COL9: Override DPD idle state of the gp3_ps[1] pad 0 = DISABLE 1 = ENABLE
8	0x0	KBC_COL8: Override DPD idle state of the gp3_ps[0] pad 0 = DISABLE 1 = ENABLE
7	0x0	KBC_COL7: Override DPD idle state of the gp3_pr[7] pad 0 = DISABLE 1 = ENABLE
6	0x0	KBC_COL6: Override DPD idle state of the gp3_pr[6] pad 0 = DISABLE 1 = ENABLE
5	0x0	KBC_COL5: Override DPD idle state of the gp3_pr[5] pad 0 = DISABLE 1 = ENABLE
4	0x0	KBC_COL4: Override DPD idle state of the gp3_pr[4] pad 0 = DISABLE 1 = ENABLE
3	0x0	KBC_COL3: Override DPD idle state of the gp3_pr[3] pad 0 = DISABLE 1 = ENABLE
2	0x0	KBC_COL2: Override DPD idle state of the gp3_pr[2] pad 0 = DISABLE 1 = ENABLE
1	0x0	KBC_COL1: Override DPD idle state of the gp3_pr[1] pad 0 = DISABLE 1 = ENABLE
0	0x0	KBC_COL0: Override DPD idle state of the gp3_pr[0] pad 0 = DISABLE 1 = ENABLE

9.2.9 APBDEV_PMC_DPD_SAMPLE_0

Setting this register will trigger sampling pads data and direction in which the pad will be driven during Deep Sleep mode.

Before writing to this register, all interfaces going to pads must be set to the ideal "idle" mode, which is expected to be driven by pads when the Tegra K1 chip enters Deep Sleep.

DPS Power Down Sample has to precede Deep Power Down Enable write. The DPD sample should not be deasserted until the transition from Deep Sleep to WB0.

This affects only for MPIO pads.

Note: Only one sample signal exists to perform sampling while entering DPD mode. Thus all interfaces are sampled at the same time. The sequence of multiple DPD groups requires keeping all interfaces (already in DPD mode) in idle.

Deep Power Down Sample

Offset: 0x20 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ON: Sets the sampling of pads value 0 = DISABLE 1 = ENABLE

9.2.10 APBDEV_PMC_DPD_ENABLE_0

Setting this register will trigger entering Deep Sleep state. It must be preceded by a DPD_SAMPLE write.

Will cause request for shutting down the system clock and the core req power. Puts the PLLs and I/Os in Deep Power Down mode. Will cut off (clamp) all signals going from core power to AO and pads.

If none of the wake-up events is set, only power-on reset can re-enable access to the chip.

After servicing of the wake-up event is completed, the register should be set back to 0 to complete a Deep Sleep cycle.

Deep Power Down Enable

Offset: 0x24 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	TSC_MULT_EN: TSC multiplier enable. 0: DISABLE (i.e., counter runs at oscillator clock and increments by 1 every clock cycle). 1: ENABLE (i.e., counter runs at 32 kHz and increments by TSC_MULT_VALUE every clock cycle). 0 = DISABLE 1 = ENABLE
0	0x0	ON: Sets sampling of pads value 0 = DISABLE 1 = ENABLE

9.2.11 APBDEV_PMC_PWRGATE_TIMER_OFF_0

Specifies the number of APB cycles after which the rail line goes off (turns the power to part of power-gated partition). Each rail controls part of the powered partition. This register should be set before the write to Power Gate Toggle. Shared between all power partitions.

Power Gate Timer Off Register

Offset: 0x28 │ Read/Write: R/W │ Reset: 0xedcba987 (0b11101101110010111010100110000111)

Bit	Reset	Description
31:28	0xe	RAIL7: Timer value for rail 7
27:24	0xd	RAIL6: Timer value for rail 6

Bit	Reset	Description
23:20	0xc	RAIL5: Timer value for rail 5
19:16	0xb	RAIL4: Timer value for rail 4
15:12	0xa	RAIL3: Timer value for rail 3
11:8	0x9	RAIL2: Timer value for rail 2
7:4	0x8	RAIL1: Timer value for rail 1
3:0	0x7	RAIL0: Timer value for rail 0

9.2.12 APBDEV_PMC_CLAMP_STATUS_0

This register is kept for backward code compatibility; the timer is based on PWRGATE_TIMER_OFF only.

Offset: 0x2c │ Read/Write: RO │ Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24	X	IRAM: Clamp status of IRAM partition 0 = DISABLE 1 = ENABLE
23	X	VIC: Clamp status of VIC partition 0 = DISABLE 1 = ENABLE
22	X	XUSBC: Clamp status of XUSBC partition 0 = DISABLE 1 = ENABLE
21	X	XUSBB: Clamp status of XUSBB partition 0 = DISABLE 1 = ENABLE
20	X	XUSBA: Clamp status of XUSBA partition 0 = DISABLE 1 = ENABLE
19	X	DISB: Clamp status of DISB partition 0 = DISABLE 1 = ENABLE
18	X	DIS: Clamp status of DIS partition 0 = DISABLE 1 = ENABLE
17	X	SOR: Clamp status of SOR partition 0 = DISABLE 1 = ENABLE
16	X	C1NC: Clamp status of cluster1 nENABLE CPU partition 0 = DISABLE 1 = ENABLE
15	X	C0NC: Clamp status of cluster0 nENABLE CPU partition 0 = DISABLE 1 = ENABLE
14	X	CE0: Clamp status of CE0 partition 0 = DISABLE 1 = ENABLE
12	X	CELP: Clamp status of CELP partition 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	X	CE3: Clamp status of CE3 partition 0 = DISABLE 1 = ENABLE
10	X	CE2: Clamp status of CE2 partition 0 = DISABLE 1 = ENABLE
9	X	CE1: Clamp status of CE1 partition 0 = DISABLE 1 = ENABLE
8	X	SAX: Clamp status of SAX partition 0 = DISABLE 1 = ENABLE
7	X	HEG: Clamp status of HEG partition 0 = DISABLE 1 = ENABLE
6	X	MPE: Clamp status of MPE partition 0 = DISABLE 1 = ENABLE
4	X	VDE: Clamp status of VDE partition 0 = DISABLE 1 = ENABLE
3	X	PCX: Clamp status of PCX partition 0 = DISABLE 1 = ENABLE
2	X	VE: Clamp status of VE partition 0 = DISABLE 1 = ENABLE
1	X	TD: Clamp status of TD partition 0 = DISABLE 1 = ENABLE
0	X	CRAIL: Clamp status of CPU Rail 0 = DISABLE 1 = ENABLE

9.2.13 APBDEV_PMC_PWRGATE_TOGGLE_0

Write to this register will turn power on/off to the specified power-gated partition. Only one partition is turned on/off at a time.

PWRGATE_STATUS should be read to determine the state of the partition before writing to this register.

Turning the partition off will cause automatic clamping of all signals generated by the power-gated partition being turned off. Before the partition is turned off, all clocks to the partition should be stopped, and the reset to the partition should be asserted.

Turning the partition on will not remove clamping. Clamping is removed only after a REMOVE_CLAMPING_CMD write.

The user must allow a minimum 20 APB clock cycles between consecutive partition Power-Gate Toggle requests.

The role of the START bit has changed from prior Tegra devices. The START bit is cleared by hardware when the PMC accepts the request to power-gate or unpower-gate the partition. So in order to power-gate/unpower-gate a partition, software needs to do the following:

- Check to see if the partition is already in the correct state, by looking at the PWRGATE_STATUS register.
- If the partition is not in the correct state, software reads the PWRGATE_TOGGLE register to see if the START bit is 0.
- If the START bit is not 0, software polls until the start bit is set to 0.

- Then program the PWRGATE_TOGGLE register with the START bit set to 1 and choose the required partition to be power-gated.
- Ideally, software can poll to check the START bit going back to 0, which indicates that the PMC has accepted the request, and then poll the STATUS register to make sure the required partition is power-gated/unpower-gated.

Power Gate Toggle

Offset: 0x30 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxx00000)

Bit	Reset	Description
8	0x0	START: Start power down/up 0 = DISABLE 1 = ENABLE
4:0	0x0	PARTID: ID of the partition to be toggled 0 = CRAIL 1 = TD 2 = Video Encode 3 = PCX 4 = Video Decode 5 = L2 Cache 6 = MPEG Encode 7 = HEG 8 = SAX 9 = CE1 10 = CE2 11 = CE3 12 = CELP 14 = CE0 15 = C0NC 16 = C1NC 17 = SOR 18 = DIS 19 = DISB 20 = XUSBA 21 = XUSBB 22 = XUSBC 23 = VIC 24 = IRAM

9.2.14 APBDEV_PMC_REMOVE_CLAMPING_CMD_0

This is a bitmap with one bit per power partition controller by the PMC.

When written to 1b, the PMC removes the clamp signals to the corresponding partition. If the partition is not powered on, the register write will be ignored.

The bit is automatically reset to 0b when the clamping has been removed. Software is responsible for writing to this register at the correct time, that is, when the clocks are started but the blocks in that partition are still held in reset.

Remove Clamping

Offset: 0x34 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxx00000000000x0000000000000)

Bit	Reset	Description
24	0x0	IRAM: Remove clamping from IRAM partition 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
23	0x0	VIC: Remove clamping from VIC partition 0 = DISABLE 1 = ENABLE
22	0x0	XUSBC: Remove clamping from XUSBC partition 0 = DISABLE 1 = ENABLE
21	0x0	XUSBB: Remove clamping from XUSBB partition 0 = DISABLE 1 = ENABLE
20	0x0	XUSBA: Remove clamping from XUSBA partition 0 = DISABLE 1 = ENABLE
19	0x0	DISB: Remove clamping from DISB partition 0 = DISABLE 1 = ENABLE
18	0x0	DIS: Remove clamping from DIS partition 0 = DISABLE 1 = ENABLE
17	0x0	SOR: Remove clamping from SOR partition 0 = DISABLE 1 = ENABLE
16	0x0	C1NC: Remove clamping from cluster 1 non-CPU partition 0 = DISABLE 1 = ENABLE
15	0x0	C0NC: Remove clamping from cluster 0 non-CPU partition 0 = DISABLE 1 = ENABLE
14	0x0	CE0: Remove clamping from CE0 partition 0 = DISABLE 1 = ENABLE
12	0x0	A9LP: Remove clamping from CELP partition 0 = DISABLE 1 = ENABLE
11	0x0	CE3: Remove clamping from CE3 partition 0 = DISABLE 1 = ENABLE
10	0x0	CE2: Remove clamping from CE2 partition 0 = DISABLE 1 = ENABLE
9	0x0	CE1: Remove clamping from CE1 partition 0 = DISABLE 1 = ENABLE
8	0x0	SAX: Remove clamping from SAX partition 0 = DISABLE 1 = ENABLE
7	0x0	HEG: Remove clamping from HEG partition 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	0x0	MPE: Remove clamping from MPE_CACHE partition 0 = DISABLE 1 = ENABLE
5	0x0	L2C: Remove clamping from L2_CACHE partition 0 = DISABLE 1 = ENABLE
4	0x0	PCX: Remove clamping from PCX partition 0 = DISABLE 1 = ENABLE
3	0x0	VDE: Remove clamping from VDE partition 0 = DISABLE 1 = ENABLE
2	0x0	VE: Remove clamping from VE partition 0 = DISABLE 1 = ENABLE
1	0x0	TD: Remove clamping from TD partition 0 = DISABLE 1 = ENABLE
0	0x0	CRAIL: Remove clamping from CPU Rail domain 0 = DISABLE 1 = ENABLE

9.2.15 APBDEV_PMC_PWRGATE_STATUS_0

This read-only register displays the status of the power partitions. This register should be read before writing to PWRGATE_TOGGLE.

Power Gate Status

Offset: 0x38 │ Read/Write: RO │ Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24	X	IRAM: Status of IRAM partition 0 = OFF 1 = ON
23	X	VIC: Status of VIC partition 0 = OFF 1 = ON
22	X	XUSBC: Status of XUSBC partition 0 = OFF 1 = ON
21	X	XUSBB: Status of XUSBB partition 0 = OFF 1 = ON
20	X	XUSBA: Status of XUSBA partition 0 = OFF 1 = ON

Bit	Reset	Description
19	X	DISB: Status of DISB partition 0 = OFF 1 = ON
18	X	DIS: Status of DIS partition 0 = OFF 1 = ON
17	X	SOR: Status of SOR partition 0 = OFF 1 = ON
16	X	C1NC: Status of cluster1 non-CPU partition 0 = OFF 1 = ON
15	X	C0NC: Status of cluster0 non-CPU partition 0 = OFF 1 = ON
14	X	CE0: Status of CE0 partition 0 = OFF 1 = ON
12	X	CELP: Status of CELP partition 0 = OFF 1 = ON
11	X	CE3: Status of CE3 partition 0 = OFF 1 = ON
10	X	CE2: Status of CE2 partition 0 = OFF 1 = ON
9	X	CE1: Status of CE1 partition 0 = OFF 1 = ON
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

Bit	Reset	Description
2	X	VE: Status of VE partition 0 = OFF 1 = ON
1	X	TD: Status of TD partition 0 = OFF 1 = ON
0	X	CRAIL: Status of CPU Rail 0 = OFF 1 = ON

9.2.16 APBDEV_PMC_PWRGOOD_TIMER_0

This register programs the length of the wake-up reset, asserted after wake-up from Deep Sleep. This register should be set before entering Deep Sleep mode.

Programmed values depend on the properties of the PMIC.

Timer value x 30.51 µs = reset pulse length

Power Good Timer

Offset: 0x3c │ Read/Write: R/W │ Reset: 0x003f007f (0bxxxxxxxx00111111000000001111111)

Bit	Reset	Description
23:16	0x3f	OSC_PREPWR: OSC clock stabilization timer prior to SoC rail pwr-req assertion. The timer value is 4*OSC_PREPWR+1 number of 32.768 kHz clock cycles; that is, the timer value is (OSC_PREPWR*122.07)+30.518 µs.
15:8	0x0	OSC_POSTPWR: OSC clock stabilization timer after SoC rail power is stabilized. The timer value is 4*OSC_POSTPWR+1 number of 32.768 kHz clock cycles; that is, the timer value is (OSC_POSTPWR*122.07)+30.518 µs.
7:0	0x7f	PWRGOOD: SoC rail power-on stabilization timer.The timer value is PWRGOOD+1 number of 32.768 kHz clock cycles; that is, the timer value is (PWRGOOD*30.52)+30.518 µs.

9.2.17 APBDEV_PMC_BLINK_TIMER_0

Will output value to pad only if the PMC Control register has BLINK_EN set and DPD_PADS_ORIDE has blink bit set.

Setting bit 15 to 1 will output 32 kHz clock (the registers above still have to be set)

- On time DATA_ON << 2 x 30.51 µs
- Off time DATA_OFF << 2 x 30.51 µs.

Blinker Timer for External Blinker

Offset: 0x40 │ Read/Write: R/W │ Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:16	0xffff	DATA_OFF: Time off
15	0x1	FORCE_BLINK: if 0 32 kHz clock
14:0	0x7fff	DATA_ON: Time on

9.2.18 APBDEV_PMC_NO_IOPOWER_0

IMPORTANT: Before 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.

No I/O Power Register

Offset: 0x44 │ Read/Write: R/W │ Reset: 0x00010080 (0bxxxxxxxxxxxxxxx010000000x1x000000)

Bit	Reset	Description
17	0x0	SYS_2:rail AO I/Os 0 = DISABLE 1 = ENABLE
16	0x1	MEM_COMP:MEM0 ADDR1 (comp cell I/Os) 0 = DISABLE 1 = ENABLE
15	0x0	HV: rail HV I/Os 0 = DISABLE 1 = ENABLE
14	0x0	SDMMC4: rail SDMMC4 I/Os 0 = DISABLE 1 = ENABLE
13	0x0	SDMMC3: rail SDMMC3 I/Os 0 = DISABLE 1 = ENABLE
12	0x0	SDMMC1: rail SDMMC1 I/Os 0 = DISABLE 1 = ENABLE
11	0x0	PEX_CNTRL: PEX 0 = DISABLE 1 = ENABLE
10	0x0	CAM: Cam rail I/Os 0 = DISABLE 1 = ENABLE
9	0x0	MIPI: MIPI rail I/Os 0 = DISABLE 1 = ENABLE
7	0x1	MEM: rail memory I/Os 0 = DISABLE 1 = ENABLE
5	0x0	AUDIO: rail I2S I/Os 0 = DISABLE 1 = ENABLE
4	0x0	VI: Defunct DVI I/Os 0 = DISABLE 1 = ENABLE
3	0x0	BB: rail DLCD I/Os 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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

9.2.19 APBDEV_PMC_PWR_DET_0

Active high, sets power detection for nine power rails only. - MIPI does not have power detect.

The proper sequence for turning on power detects:

- Write 1 to PWR_DET register for fields requiring power detect
- Allow for 3 µs delay (in APB clock cycles)
- Write 1 to PWR_DET_LATCH

For turning PWR_DET off:

- Write 0 to PWR_DET_LATCH
- No delay is necessary
- Write 0 to PWR_DET

Power Detect

Offset: 0x48 │ Read/Write: R/W │ Reset: 0x0002bc2f (0bxxxxxxxxxxxxxxx1x1x1111xxxx1x1111)

Bit	Reset	Description
17	0x1	SYS_2:rail AO I/Os 0 = DISABLE 1 = ENABLE
15	0x1	HV:HV rail 0 = DISABLE 1 = ENABLE
13	0x1	SDMMC3: rail SDMMC3 I/Os 0 = DISABLE 1 = ENABLE
12	0x1	SDMMC1: rail SDMMC1 I/Os 0 = DISABLE 1 = ENABLE
11	0x1	PEX_CNTRL: PEX rail 0 = DISABLE 1 = ENABLE
10	0x1	CAM: rail Cam I/Os 0 = DISABLE 1 = ENABLE
5	0x1	AUDIO: rail I2S I/Os 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
3	0x1	BB:rail DLCD I/Os 0 = DISABLE 1 = ENABLE
2	0x1	UART: rail DBG I/Os 0 = DISABLE 1 = ENABLE
1	0x1	NAND: rail GMI I/Os 0 = DISABLE 1 = ENABLE
0	0x1	SYS: rail AO I/Os 0 = DISABLE 1 = ENABLE

9.2.20 APBDEV_PMC_PWR_DET_LATCH_0

Latches power detect for power rails enabled by Power Detect register.

Power Detect Latch

Offset: 0x4c │ Read/Write: R/W │ Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	LATCH: power detect latch, latches value as long set to 1 0 = DISABLE 1 = ENABLE

9.2.21 APBDEV_PMC_SCRATCH0_0

Scratch Register

Scratch registers for restoring context after wake-up. On a cold power up, the content of register 0 will be reset to 0x0.

Offset: 0x50 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH0: General-purpose register storage

9.2.22 APBDEV_PMC_SCRATCH1_0

Scratch Register

Offset: 0x54 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH1: General-purpose register storage

9.2.23 APBDEV_PMC_SCRATCH2_0

Scratch Register

Offset: 0x58 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH2: General-purpose register storage

9.2.24 APBDEV_PMC_SCRATCH3_0

Scratch Register

Offset: 0x5c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH3: General-purpose register storage

9.2.25 APBDEV_PMC_SCRATCH4_0

Scratch Register

Offset: 0x60 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH4: General-purpose register storage

9.2.26 APBDEV_PMC_SCRATCH5_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH5: General-purpose register storage

9.2.27 APBDEV_PMC_SCRATCH6_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH6: General-purpose register storage

9.2.28 APBDEV_PMC_SCRATCH7_0

Scratch Register

Offset: 0x6c │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH7: General-purpose register storage

9.2.29 APBDEV_PMC_SCRATCH8_0

Scratch Register

Offset: 0x70 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH8: General-purpose register storage

9.2.30 APBDEV_PMC_SCRATCH9_0

Scratch Register

Offset: 0x74 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH9: General-purpose register storage

9.2.31 APBDEV_PMC_SCRATCH10_0

Scratch Register

Offset: 0x78 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH10: General-purpose register storage

9.2.32 APBDEV_PMC_SCRATCH11_0

Scratch Register

Offset: 0x7c │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH11: General-purpose register storage

9.2.33 APBDEV_PMC_SCRATCH12_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH12: General-purpose register storage

9.2.34 APBDEV_PMC_SCRATCH13_0

Scratch Register

Offset: 0x84 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH13: General-purpose register storage

9.2.35 APBDEV_PMC_SCRATCH14_0

Scratch Register

Offset: 0x88 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH14: General-purpose register storage

9.2.36 APBDEV_PMC_SCRATCH15_0

Scratch Register

Offset: 0x8c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH15: General-purpose register storage

9.2.37 APBDEV_PMC_SCRATCH16_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH16: General-purpose register storage

9.2.38 APBDEV_PMC_SCRATCH17_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH17: General-purpose register storage

9.2.39 APBDEV_PMC_SCRATCH18_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH18: General-purpose register storage

9.2.40 APBDEV_PMC_SCRATCH19_0

Scratch Register

Offset: 0x9c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH19: General-purpose register storage

9.2.41 APBDEV_PMC_SCRATCH20_0

Scratch Register

Offset: 0xa0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH20: General-purpose register storage

9.2.42 APBDEV_PMC_SCRATCH21_0

Scratch Register

Offset: 0xa4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH21: General-purpose register storage

9.2.43 APBDEV_PMC_SCRATCH22_0

Scratch Register

Offset: 0xa8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH22: General-purpose register storage

9.2.44 APBDEV_PMC_SCRATCH23_0

Scratch Register

Offset: 0xac | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH23: General-purpose register storage

9.2.45 APBDEV_PMC_SECURE_SCRATCH0_0

Secure Scratch Register

Offset: 0xb0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH0

9.2.46 APBDEV_PMC_SECURE_SCRATCH1_0

Secure Scratch Register

Offset: 0xb4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH1

9.2.47 APBDEV_PMC_SECURE_SCRATCH2_0

Secure Scratch Register

Offset: 0xb8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH2

9.2.48 APBDEV_PMC_SECURE_SCRATCH3_0

Secure Scratch Register

Offset: 0xbc │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH3

9.2.49 APBDEV_PMC_SECURE_SCRATCH4_0

Secure Scratch Register

Offset: 0xc0 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH4

9.2.50 APBDEV_PMC_SECURE_SCRATCH5_0

Secure Scratch Register

Offset: 0xc4 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH5

9.2.51 APBDEV_PMC_CPUPWRGOOD_TIMER_0

CPU Power Good Timer

The behavior for the DATA timers 0 to 2 is the same (i.e., 2 cycles of delay).

This timer runs on the APB clock.

It should also be used to mask OC during CPU rail power up when OC is multiplexed to the PG pin using APBDEV_PMC_CNTRL_0_CPUPWRGOOD_EN based on how long the PMIC drives PG on this signal.

Offset: 0xc8 │ Read/Write: R/W │ Reset: 0x0000ffff (0b00000000000000001111111111111111)

Bit	Reset	Description
31:0	0xffff	DATA: Timer data

9.2.52 APBDEV_PMC_CPUPWROFF_TIMER_0

CPU Power Off Timer

Used for On-to-Off transitions.

This timer runs on the APB clock

It should also be used to mask OC during CPU rail power down when OC is multiplexed to the PG pin using APBDEV_PMC_CNTRL_0_CPUPWRGOOD_EN based on how long the PMIC keeps PG low.

Note: This timer must always have a non-zero value.

Offset: 0xcc │ Read/Write: R/W │ Reset: 0x0000ffff (0b00000000000000001111111111111111)

Bit	Reset	Description
31:0	0xffff	DATA: Timer data

9.2.53 APBDEV_PMC_PG_MASK_0

Offset: 0xd0 │ Read/Write: R/W │ Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:24	0xff	PCX: Mask PCX rail
23:16	0xff	VD: Mask VDE rail
15:8	0xff	VE: Mask VE rail
7:0	0xff	TD: Mask TD rail

9.2.54 APBDEV_PMC_PG_MASK_1_0

Offset: 0xd4 │ Read/Write: R/W │ Reset: 0xffffff01 (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

9.2.55 APBDEV_PMC_AUTO_WAKE_LVL_0

Note: This register is not used.

The wake levels are snapped just before entering DPD by default.

Offset: 0xd8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	SMPL: Causes PMC to sample the wake pads 0 = DISABLE 1 = ENABLE

9.2.56 APBDEV_PMC_AUTO_WAKE_LVL_MASK_0

This register is used by software to enable sampling of the wake pads before entering Deep Sleep.

Setting a '1' causes the associated pad to be sampled, and the value transferred to the WAKE_LVL register.

Offset: 0xdc │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE

9.2.57 APBDEV_PMC_WAKE_DELAY_0

WAKE_DELAY should be a non-zero value.

Offset: 0xe0 │ Read/Write: R/W │ Reset: 0x00000001 (0bxxxxxxxxxxxxxxxx0000000000000001)

Bit	Reset	Description
15:0	0x1	VALUE

9.2.58 APBDEV_PMC_PWR_DET_VAL_0

This register is used to override the power-detect cells and manually set the values (in the unlikely case the power-detect cells are broken). A write to this register also causes the values from the power-detect cells to be captured and which can be subsequently read out.

Offset: 0xe4 │ Read/Write: R/W │ Reset: 0x0002bc2f (0bxxxxxxxxxxxxxxx1x1x1111xxxx1x1111)

Bit	Reset	Description
17	0x1	SYS_2: rail AO I/Os 0 = ENABLE 1 = DISABLE
15	0x1	HV:rail HV I/Os 0 = ENABLE 1 = DISABLE
13	0x1	SDMMC3: rail SDMMC3 I/Os 0 = ENABLE 1 = DISABLE
12	0x1	SDMMC1: rail SDMMC1 I/Os 0 = ENABLE 1 = DISABLE
11	0x1	PEX_CNTRL: PEX rail 0 = ENABLE 1 = DISABLE
10	0x1	CAM: rail cam I/Os 0 = ENABLE 1 = DISABLE
5	0x1	AUDIO: rail I2S I/Os 0 = ENABLE 1 = DISABLE
3	0x1	BB: rail DLCD I/Os 0 = ENABLE 1 = DISABLE
2	0x1	UART: rail DBG I/Os 0 = ENABLE 1 = DISABLE
1	0x1	NAND: rail AT3 I/Os

Bit	Reset	Description
		0 = ENABLE 1 = DISABLE
0	0x1	SYS: rail AO I/Os 0 = ENABLE 1 = DISABLE

9.2.59 APBDEV_PMC_DDR_PWR_0

This register is used to program the "E_18V" pin of the DDR pads.

Offset: 0xe8 │ Read/Write: R/W │ Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx11)

Bit	Reset	Description
1	0x1	EMMC: GMI pins 0 = E_12V 1 = E_18V
0	0x1	VAL: DVI pins 0 = E_12V 1 = E_18V

9.2.60 APBDEV_PMC_USB_DEBOUNCE_DEL_0

Determines number of 32 kHz clock cycles to debounce USB signal events.

Note: The programmed debounce values must be greater than 1.

Reset values are for Cold Hardware Reset only.

Offset: 0xec │ Read/Write: R/W │ Reset: 0x00000002 (0bxxxxxxxx00000000000000000000010)

Bit	Reset	Description
23:20	0x0	UHSIC_LINE_DEB_CNT: Number of 32 kHz debounce cycles for UHSIC port 0
19:16	0x0	UTMIP_LINE_DEB_CNT: Number of 32 kHz debounce cycles for UTMIP port 0
15:0	0x2	VAL: Debounce period for ID and VBUS events on all USB ports. The programmed value must be greater than 1.

9.2.61 APBDEV_PMC_USB_AO_0

Power downs for various USB features controlled by the PMC.

Each UTMIP has the following power downs for features that can lead to wake-up events.

- USBOP_VAL_PD: Power down for D+ value detector (I/O pad)
- USBON_VAL_PD: Power down for D- value detector (I/O pad)
- ID_PD: Power down for ID detector (bias pad)
- VBUS_WAKEUP_PD: Power down for VBUS detector (bias pad)

Each UHSIC port has the following power downs for features that can lead to wake-up events:

- STROBE_VAL_PD: Power down for STROBE VALUE detector
- DATA_VAL_PD: Power down for DATA VALUE detector

Note: These HSIC pins have not been added to the pad yet.

Line Debounce Periods for UTMIP and HSIC PORTS

Offset: 0xf0 | Read/Write: R/W | Reset: 0x000fffff (0bxxxxxxxxxxxx11111111111111111111)

Bit	Reset	Description
19:18	0x3	UHSIC_RESERVED_P1: 2 bits reserved for UHSIC P1
17	0x1	DATA_VAL_PD_P1: Power Down D- DATA_VAL receiver for UHSIC P1
16	0x1	STROBE_VAL_PD_P1: Power Down D+ STROBE_VAL receiver for UHSIC P1
15:14	0x3	UHSIC_RESERVED_P0: 2 bits reserved for UHSIC P1
13	0x1	DATA_VAL_PD_P0: Power Down D- DATA_VAL receiver for UHSIC P0
12	0x1	STROBE_VAL_PD_P0: Power Down D+ STROBE_VAL receiver for UHSIC P0
11	0x1	ID_PD_P2: Power Down ID Wake up for UTMIP
10	0x1	VBUS_WAKEUP_PD_P2: Power Down Vbus Wake Up for UTMIP P2
9	0x1	USBON_VAL_PD_P2: Power Down D- USBOP_VAL receiver for UTMIP P0
8	0x1	USBOP_VAL_PD_P2: Power Down D+ USBOP_VAL receiver for UTMIP P0
7	0x1	ID_PD_P1: Power Down ID Wake up for UTMIP P1
6	0x1	VBUS_WAKEUP_PD_P1: Power Down Vbus Wake Up for UTMIP P1
5	0x1	USBON_VAL_PD_P1: Power Down D- USBOP_VAL receiver for UTMIP P0
4	0x1	USBOP_VAL_PD_P1: Power Down D+ USBOP_VAL receiver for UTMIP P0
3	0x1	ID_PD_P0: Power Down ID Wake up for UTMIP
2	0x1	VBUS_WAKEUP_PD_P0: Power Down Vbus Wake Up for UTMIP P0
1	0x1	USBON_VAL_PD_P0: Power Down D- USBOP_VAL receiver for UTMIP P0
0	0x1	USBOP_VAL_PD_P0: Power Down D+ USBOP_VAL receiver for UTMIP P0

9.2.62 APBDEV_PMC_CRYPTO_OP_0

A complete solution requires a "semi-sticky" bit in the always-on domain. The Boot ROM would clear this semi-sticky bit for cold boots, but use its value to propagate the crypto-disable flag across LP0. (The Boot ROM always requires crypto functionality, so a pure hardware solution probably isn't reasonable.)

The Boot ROM would be able to clear (to zero) and read the sticky bit, but outside the Boot ROM users would only be able to set (to one) and read the sticky bit. On cold boot, the Boot ROM would always clear the stick bit. On WB0, the Boot ROM would read the sticky bit and copy it's setting to the crypto disable flag.

Crypto Engine Disable Sticky Bit

Offset: 0xf4 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	VAL: Disabled by default 0 = ENABLE 1 = DISABLE

9.2.63 APBDEV_PMC_PLLP_WB0_OVERRIDE_0

Master control for all WB0 PLL overrides.

Offset: 0xf8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000000000000)

Bit	Reset	Description
12	0x0	PLLM_ENABLE: 1 = enable PLLM, 0 = disable PLLM
11	0x0	PLLM_OVERRIDE_ENABLE: 1 = override CAR PLLM setting, 0 = no override. PLLM WB to programmable frequency.
10	0x0	PLLU_ENABLE: 1 = enable PLLU, 0 = disable PLLU.
9	0x0	PLLU_OVERRIDE_ENABLE: 1 = override CAR PLLU setting, 0 = no override. PLLU WB to fixed frequency
8	0x0	OSC_OVERRIDE_ENABLE: 1 = override CAR OSC setting, 0 = no override. Controls OSC_FREQ and PLL_REF_DIV
7:6	0x0	PLL_REF_DIV: PLL reference clock divide for all PLLs. 00 = /1, 01 = /2, 10 = /4, 11 = reserve.
5:2	0x0	OSC_FREQ: Oscillator frequency for shared PLL reference 0000 = 13MHz 0100 = 19.2MHz 1000 = 12MHz 1100 = 26MHz 0001 = 16.8MHz 0101 = 38.4MHz 1001 = 48.0MHz
1	0x0	PLLP_ENABLE: 1 = enable PLLP, 0 = disable PLLP
0	0x0	PLLP_OVERRIDE_ENABLE: 1 = override CAR PLLP setting, 0 = no override. PLLP WB to fixed frequency.

9.2.64 APBDEV_PMC_SCRATCH24_0

Scratch Register

Offset: 0xfc │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH24: General-purpose register storage

9.2.65 APBDEV_PMC_SCRATCH25_0

Scratch Register

Offset: 0x100 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH25: General-purpose register storage

9.2.66 APBDEV_PMC_SCRATCH26_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH26: General-purpose register storage

9.2.67 APBDEV_PMC_SCRATCH27_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH27: General-purpose register storage

9.2.68 APBDEV_PMC_SCRATCH28_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH28: General-purpose register storage

9.2.69 APBDEV_PMC_SCRATCH29_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH29: General-purpose register storage

9.2.70 APBDEV_PMC_SCRATCH30_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH30: General-purpose register storage

9.2.71 APBDEV_PMC_SCRATCH31_0

Scratch Register

Offset: 0x118 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH31: General-purpose register storage

9.2.72 APBDEV_PMC_SCRATCH32_0

Scratch Register

Offset: 0x11c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH32: General-purpose register storage

9.2.73 APBDEV_PMC_SCRATCH33_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH33: General-purpose register storage

9.2.74 APBDEV_PMC_SCRATCH34_0

Scratch Register

Offset: 0x124 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH34: General-purpose register storage

9.2.75 APBDEV_PMC_SCRATCH35_0

Scratch Register

Offset: 0x128 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH35: General-purpose register storage

9.2.76 APBDEV_PMC_SCRATCH36_0

Scratch Register

Offset: 0x12c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH36: General-purpose register storage

9.2.77 APBDEV_PMC_SCRATCH37_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH37: General-purpose register storage

9.2.78 APBDEV_PMC_SCRATCH38_0

Scratch Register

Offset: 0x134 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH38: General-purpose register storage

9.2.79 APBDEV_PMC_SCRATCH39_0

Scratch Register

Offset: 0x138 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH39: General-purpose register storage

9.2.80 APBDEV_PMC_SCRATCH40_0

Scratch Register

Offset: 0x13c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH40: General-purpose register storage

9.2.81 APBDEV_PMC_SCRATCH41_0

Scratch Register

Offset: 0x140 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH41: General-purpose register storage

9.2.82 APBDEV_PMC_SCRATCH42_0

Scratch Register

Offset: 0x144 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH42: General-purpose register storage

9.2.83 APBDEV_PMC_BONDOUT_MIRROR0_0

Secure Scratch Register

Offset: 0x148 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR0

9.2.84 APBDEV_PMC_BONDOUT_MIRROR1_0

Secure Scratch Register

Offset: 0x14c │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR1

9.2.85 APBDEV_PMC_BONDOUT_MIRROR2_0

Secure Scratch Register

Offset: 0x150 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR2

9.2.86 APBDEV_PMC_SYS_33V_EN_0

Offset: 0x154 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	VAL: 1 = 3.3V 0 = 1.8V

9.2.87 APBDEV_PMC_BONDOUT_MIRROR_ACCESS_0

On separate reset (same as the CAR) disables access (read/write to secure scratch registers)

Offset: 0x158 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	BREAD: disable read from bondout secure registers 0 = OFF 1 = ON
0	0x0	BWRITE: disable write to bondout secure registers 0 = OFF 1 = ON

9.2.88 APBDEV_PMC_GATE_0

This register is used for software controlled synchronization between APB domain and the 32 kHz domain. Software is expected to set the GAKE_WAKE/GATE_DBNS field high to cut the 32 kHz gated clock before updating WAKE_LVL, AUTO_WAKE_LVL, WAKE_MASK and USB_DEBOUNCE_DEL registers. After these registers have been written the GATE_WAKE/GATE_DBNS bit should be written back to '0' to enable the 32 kHz gated clock.

This gates the 32 kHz clock to just the flops that store the above fields.

Offset: 0x15c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	GATE_WAKE: 0 = OFF 1 = ON

Bit	Reset	Description
0	0x0	GATE_DBNS: 0 = OFF 1 = ON

9.2.89 APBDEV_PMC_WAKE2_MASK_0

Auto-wake is not present for wake2 events in Tegra K1 devices.

The APBDEV_PMC_WAKE2 registers handle wake events whose number exceeds 32. For wake events whose number is less than or equal to 32, refer to the APBDEV_PMC_WAKE registers above.

Wake2 alignments for masks, level, and status:

- WAKE2_*[0] = ulpi_data3 (wake event 32)
- WAKE2_*[1] = gpio_pj[0] (wake event 33)
- WAKE2_*[2] = gpio_pk[2] (wake event 34)
- WAKE2_*[3] = gpio_pi[6] (wake event 35)
- WAKE2_*[7]= utmip0_line_wakup_event (wake event 39)
- WAKE2_*[8]= utmip1_line_wakup_event (wake event 40)
- WAKE2_*[9]= utmip2_line_wakup_event (wake event 41)
- WAKE2_*[10]= uhsic_line_wakup_event (wake event 42)
- WAKE2_*[11]= uhsic2_line_wakup_event (wake event 43)
- WAKE2_*[12]= gen1_i2c1_sda (wake event 44)
- WAKE2_*[13]= gpio_pbb6 (wake event 45)
- WAKE2_*[14]= pwr_i2c_sda (wake event 46)
- WAKE2_*[15]= gen2_i2c_sda (wake event 47)
- WAKE2_*[16]= cam_i2c_sda (wake event 48)
- WAKE2_*[17]= kb_row7 (wake event 49)
- WAKE2_*[18]= kb_row4 (wake event 50)
- WAKE2_*[19]= kb_col0 (wake event 51)
- WAKE2_*[20]= hdmi_cec (wake event 52)
- WAKE2_*[21]= cam_i2c_scl (wake event 53)
- WAKE2_*[22]= kb_col5 (wake event 54)
- WAKE2_*[23]= uart3_cts_n (wake event 55)
- WAKE2_*[24]= sdmmc3_cd_n (wake event 56)
- WAKE2_*[25]= spdif_in (wake event 57)
- WAKE2_*[26] = wake2pmc_xusb_system_wakeup (wake event 58)

Offset: 0x160 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxx00000000000000000000000000000)

Bit	Reset	Description
28:12	0x0	EVENT_REST_1: 0 = DISABLE 1 = ENABLE
11:7	0x0	EVENT_REST: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6:5	0x0	USB_EVENT: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
4:0	0x0	EVENT: 0 = DISABLE 1 = ENABLE

9.2.90 APBDEV_PMC_WAKE2_LVL_0

This register sets the active level for a wake event. It causes an exit from the Deep Sleep state if the input signal level matches the level set in this register and if WAKE2_MASK is set for the event to 1. A level is not needed for 4 line wakeup events; the level is always 1.

Set the following PMC register bits to '1' to set the wake signal active level to 'HIGH':

- APBDEV_PMC_WAKE2_LVL_0[5] for VBUS wakeup
- APBDEV_PMC_WAKE2_LVL_0[6] for ID wakeup
- APBDEV_PMC_WAKE2_LVL_0[7] for USB2.0 port 0 wakeup
- APBDEV_PMC_WAKE2_LVL_0[8] for USB2.0 port 1 wakeup
- APBDEV_PMC_WAKE2_LVL_0[9] for USB2.0 port 2 wakeup
- APBDEV_PMC_WAKE2_LVL_0[10] for HSIC port 0 wakeup
- APBDEV_PMC_WAKE2_LVL_0[11] for HSIC port 1 wakeup
- APBDEV_PMC_WAKE2_LVL_0[26] for USB3.0 ports wakeup

PMC Wake Level

Offset: 0x164 │ Read/Write: R/W │ Reset: 0x19ffffe7 (0b00011001111111111111111111100111)

Bit	Reset	Description
31	0x0	ALLOW_PULSE_WAKE: Reserved
30:29	0x0	FUTURE_CNTL: Reserved
28:12	0x19fff	EVENT_REST_1: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
11:7	0x1f	EVENT_REST: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
6:5	0x3	USB_EVENT: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
4:0	0x7	EVENT: Pin 0-2 wake level 0 = ACTIVE_LOW 1 = ACTIVE_HIGH

9.2.91 APBDEV_PMC_WAKE2_STATUS_0

This register stores status of the wake events. An event is set if the level matches and is not masked.

Writes will reset the set wake event.

PMC Wake Status

Offset: 0x168 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxx00000000000000000000000000000)

Bit	Reset	Description
28:12	0x0	EVENT_REST_1: 0 = NOT_SET 1 = SET
11:7	0x0	EVENT_REST: Internally set USB events 0 = NOT_SET 1 = SET
6:5	0x0	USB_EVENT: USB wake events 0 = NOT_SET 1 = SET
4:0	0x0	EVENT: Pin 0-2 wake enable 0 = NOT_SET 1 = SET

9.2.92 APBDEV_PMC_SW_WAKE2_STATUS_0

This register stores the status of the wake events. Latching of the events in software wake status is enabled by the PMC_CNTRL register bit LATCHWAKE_EN. Latching will stop at a 1-to-0 transition on this bit.

An event is set if the level matches. Masking does not affect this register. Writes will reset the set wake events.

PMC Software Wake Status

Offset: 0x16c │Read/Write: R/W │ Reset: 0x00000000 (0bxxx00000000000000000000000000000)

Bit	Reset	Description
28:12	0x0	EVENT_REST_1: 0 = NOT_SET 1 = SET
11:7	0x0	EVENT_REST
6:5	0x0	USB_EVENT: USB wake events 0 = NOT_SET 1 = SET
4:0	0x0	EVENT: pin 0-2 wake 0 = DISABLE 1 = ENABLE

9.2.93 APBDEV_PMC_AUTO_WAKE2_LVL_MASK_0

Auto-wake is not present for wake2 events in Tegra K1 devices.

Offset: 0x170 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxx00000000000000000000000000000)

Bit	Reset	Description
28:12	0x0	EVENT_REST_1
11:7	0x0	EVENT_REST
6:0	0x0	VALUE

9.2.94 APBDEV_PMC_PG_MASK_2_0

Power-Gate mask registers for power-gated fields. Need 32 bits for CPU.

Offset: 0x174 │ Read/Write: R/W │ Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description

Bit	Reset	Description
31:24	0xff	IRAM: Mask IRAM rail
23:16	0xff	VIC: Mask CELP rail
15:8	0xff	CELP: Mask CELP rail
7:0	0xff	TD2: Mask TD2 rail - Defunct

9.2.95 APBDEV_PMC_PG_MASK_CE1_0

Offset: 0x178 | Read/Write: R/W | Reset: 0x000000ff (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx11111111)

Bit	Reset	Description
7:0	0xff	MASK: Mask CE1 rail

9.2.96 APBDEV_PMC_PG_MASK_CE2_0

Offset: 0x17c | Read/Write: R/W | Reset: 0x000000ff (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx11111111)

Bit	Reset	Description
7:0	0xff	MASK: Mask CE1 rail

9.2.97 APBDEV_PMC_PG_MASK_CE3_0

Offset: 0x180 | Read/Write: R/W | Reset: 0x000000ff (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx11111111)

Bit	Reset	Description
7:0	0xff	MASK: Mask CE1 rail

9.2.98 APBDEV_PMC_PWRGATE_TIMER_CE_0_0

Separate power_gate_off, on, for CE counters - 4 bits each.

Offset: 0x184 | Read/Write: R/W | Reset: 0xedcba987 (0b11101101110010111010100110000111)

Bit	Reset	Description
31:28	0xe	RAIL7: Timer Value for rail 7
27:24	0xd	RAIL6: Timer Value for rail 6
23:20	0xc	RAIL5: Timer Value for rail 5
19:16	0xb	RAIL4: Timer Value for rail 4
15:12	0xa	RAIL3: Timer Value for rail 3
11:8	0x9	RAIL2: Timer Value for rail 2
7:4	0x8	RAIL1: Timer Value for rail 1
3:0	0x7	RAIL0: Timer Value for rail 0

9.2.99 APBDEV_PMC_PWRGATE_TIMER_CE_1_0

PWRGATE_TIMER CE_1 removed but kept for backward compatibility.

Offset: 0x188 | Read/Write: R/W | Reset: 0x1240e30a (0bxx01001001000000111000110000 1010)

Bit	Reset	Description

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

9.2.100 APBDEV_PMC_PWRGATE_TIMER_CE_2_0

PWRGATE_TIMER CE_2 removed but kept for backward compatibility.

Offset: 0x18c │ Read/Write: R/W │ Reset: 0x1c698594 (0bxx011100011010011000010110010100)

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

9.2.101 APBDEV_PMC_PWRGATE_TIMER_CE_3_0

PWRGATE_TIMER CE_3 removed but kept for backward compatibility.

Offset: 0x190 │ Read/Write: R/W │ Reset: 0x2692281e (0bxx100110100100100010100000011110)

Bit	Reset	Description
29:24	0x26	RAIL19: Timer Value for rail 19
23:18	0x24	RAIL18: Timer Value for rail 18
17:12	0x22	RAIL17: Timer Value for rail 17
11:6	0x20	RAIL16: Timer Value for rail 16
5:0	0x1e	RAIL15: Timer Value for rail 15

9.2.102 APBDEV_PMC_PWRGATE_TIMER_CE_4_0

PWRGATE_TIMER CE_4 removed but kept for backward compatibility.

Offset: 0x194 │ Read/Write: R/W │ Reset: 0x30bacaa8 (0bxx110000101110101100101010101000)

Bit	Reset	Description
29:24	0x30	RAIL24: Timer Value for rail 24
23:18	0x2e	RAIL23: Timer Value for rail 23
17:12	0x2c	RAIL22: Timer Value for rail 22
11:6	0x2a	RAIL21: Timer Value for rail 21
5:0	0x28	RAIL20: Timer Value for rail 20

9.2.103 APBDEV_PMC_PWRGATE_TIMER_CE_5_0

PWRGATE_TIMER CE_5 removed but kept for backward compatibility.

Offset: 0x198 | Read/Write: R/W | Reset: 0x3ae36d32 (0bxx11101011100011011010100110010)

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

9.2.104 APBDEV_PMC_PWRGATE_TIMER_CE_6_0

PWRGATE_TIMER CE_6 removed but kept for backward compatibility.

Offset: 0x19c | Read/Write: R/W | Reset: 0x00000f3b (0bxxxxxxxxxxxxxxxxxxxxxx111100111011)

Bit	Reset	Description
11:6	0x3c	RAIL31: Timer Value for rail 31
5:0	0x3b	RAIL30: Timer Value for rail 30

9.2.105 APBDEV_PMC_PCX_EDPD_CNTRL_0

Defunct

Offset: 0x1a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	EN: when set to 1, sets the EDPD to PCX (PLL toggle) 0 = OFF 1 = ON

9.2.106 APBDEV_PMC_OSC_EDPD_OVER_0

This register can be programmed to keep the oscillator ON during LP0 mode. It cuts the latency time on LP0 wake if the oscillator pad does not have to be restarted.

When the oscillator pad is on, during LP0 mode, DSPARE, duty, strength, and bypass are controlled by the fields below.

The oscillator can be in one in four modes during LP0 - in DPD, not in DPD but not enabled, running, or running only when an external request is active.

Offset: 0x1a4 | Read/Write: R/W | Reset: 0x0000007e (0bxxxxxxxx00000000000000001111110)

Bit	Reset	Description
23	0x0	CLK_OK: Crystal oscillator clk_ok signal 0 = DISABLE 1 = ENABLE
22	0x0	OSC_CTRL_SELECT: select whether the CAR's OSC_CTRL or the PMC's OSC_CTRL_OVER affects the oscillator cell. This bit should always be "1" in LP0. 0 = CAR 1 = PMC
21:20	0x0	XO_LP0_MODE: control the behavior of the oscillator during LP0 (assuming OSC_CTRL_SELECT is set to PMC during LP0). Put the oscillator in DPD mode during LP0, bypass DPD, but do not enable the oscillator during LP0, force the oscillator to run during LP0, or run the oscillator when requested by an external clock request pin 0 = DPD. 1 = OFF

Bit	Reset	Description
		2 = ON 3 = EXT_REQ
19:12	0x0	OSCFI_SPARE: Crystal oscillator spare register control.
11:7	0x0	XODS: Crystal oscillator duty cycle control.
6:1	0x3f	XOFS: Crystal oscillator drive strength control.
0	0x0	XOBP: Crystal oscillator bypass enable (1 = enable bypass).

9.2.107 APBDEV_PMC_CLK_OUT_CNTRL_0

This register controls 3 clock outputs of the Tegra K1 chip. Each of the clock outputs can be in 1 of 4 modes: not running, running when request is active high, running when request is active low, or always running. The clock output when not running can be tristated, high, or low.

The clock source might be from the CAR unit (not available when in LP0 mode), osc, osc_div2, or osc_div4.

The clock source switching on dap_mclk1_out , clk3_out and clk2_out is not glitch free. If dpd_override is not set before entering LP0, the selected clock source will get latched to "0" or "1" in LP0 (which is not glitch free).dpd_override can be "0" only when external device does not need clock in LP0, and it should be in reset before PMC_SAMPLE bit is written by LP0 entry code.

The clock source should only be switched when the pad output is not being used by an external device.

When an external device requests for a clock, the first few clocks should not be used by the external device.

Offset: 0x1a8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

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:

Bit	Reset	Description
8	0x0	CLK2_ACCEPT_REQ:
7:6	0x0	CLK1_SRC_SEL: 0 = OSC_DIV1 1 = OSC_DIV2 2 = OSC_DIV3 3 = CAR
5:4	0x0	CLK1_IDLE_STATE: state of the output line before clock request is active 0 = LOW 1 = HIGH 2 = TRIS
3	0x0	CLK1_RESERVED: reserved
2	0x0	CLK1_FORCE_EN: force the clock to run regardless of ACCEPT_REQ or INVERT_REQ
1	0x0	CLK1_INVERT_REQ:
0	0x0	CLK1_ACCEPT_REQ:

9.2.108 APBDEV_PMC_SATA_PWRGT_0

This register controls the SATA PLLs

PLLE or PADPLL can be disabled/enabled by a software or hardware request.

The SATA PWRGT is reserved for backward compatibility.

Offset: 0x1ac │ Read/Write: R/W │ Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00111111)

Bit	Reset	Description
7	0x0	SW_PLL_EDPD: Defunct. The SATA PLL is put in the DPD state when this bit is set, independently of power-gating - kept only until drivers are fixed. 0 = OFF 1 = ON
6	0x0	PG_INFO: sm2sata_pg_info
5	0x1	PLLE_IDDQ_OVERRIDE_VALUE: Defunct. 0: The PLLE is powered up. 1: Software can put the PLLE in IDDQ mode by setting this bit and PLLE_IDDQ_SWCTL (default) 0 = OFF 1 = ON
4	0x1	PLLE_IDDQ_SWCTL: 0: Defunct. The PLLE is put in IDDQ mode by hardware (SATA +AFI+CAR) signals. 1: The PLLE is put in IDDQ mode by software -- default 0 = OFF 1 = ON
3	0x1	PADPHY_IDDQ_OVERRIDE_VALUE: Defunct. Shared with XUSB0. 0: The PHY is powered up. 1: Software can put the PHY in IDDQ mode by setting this bit and SATA_PADPHY_IDDQ_SWCT (default) 0 = OFF 1 = ON
2	0x1	PADPHY_IDDQ_SWCTL: Defunct. Shared with XUSB0. 0 The SATA PHY is put in IDDQ mode by hardware (SATA +AFI+CAR) signals. 1 The SATA PHY is put in IDDQ mode by software (default) 0 = OFF 1 = ON
1	0x1	PADPLL_IDDQ_OVERRIDE_VALUE: Defunct. 0: The pad PLL is powered up 1: Software can put the pad PLL in IDDQ mode by setting this bit and SATA_PADPLL_IDDQ_SWCTL (default) 0 = OFF 1 = ON

Bit	Reset	Description
0	0x1	PADPLL_IDDQ_SWCTL: Defunct. 0: The SATA pad PLL is put in IDDQ mode by hardware (SATA +AFI+CAR) signals.1: The SATA pad PLL is put in IDDQ mode by software (default) 0 = OFF 1 = ON

9.2.109 APBDEV_PMC_SENSOR_CTRL_0

For sensor shutdown and control.

The sensor control register defines chip behavior when a sensor request is triggered. It can power-gate down all CPUs if enabled. It can trigger a reset (will cause a CPU power request to go down and power-gates down all CPUs)

IMPORTANT-- to preserve RAM content, any reads/writes to scratch registers will be blocked after a sensor triggered reset.

Software must reset BLOCK_SCRATCH_WRITE to enable writes to scratch registers.

Offset: 0x1b0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	BLOCK_SCRATCH_WRITE: Reset by user, set by the PMC when entering sensor reset 0 = OFF 1 = ON
1	0x0	ENABLE_RST: Enables reset on sensor going up. 0 = OFF 1 = ON
0	0x0	ENABLE_PG: Power gates CPUs on temperature sensor going up. 0 = OFF 1 = ON

9.2.110 APBDEV_PMC_RST_STATUS_0

Simplified Reset and Reset Source

Offset: 0x1b4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2:0	0x0	RST_SOURCE: Source of reset 0 = POR 1 = WATCHDOG 2 = SENSOR 3 = SW_MAIN 4 = LP0

9.2.111 APBDEV_PMC_IO_DPD_REQ_0

Puts I/O rails in or out of DPD mode, even though the chip is not in LP0. Multiple bits are allowed to be set at the same time. No new operation will be triggered until previous one completes. Consecutive operations issued by completion time of first one will be dropped. The register is still updated.

Setting IO_DPD_REQ should be preceded by writing to the DPD_SAMPLE registers (to enable a snapshot of data to be driven) and the SEL_DPD_TIM register. SEL_DPD_TIM is set to a safe value, but it can be lowered if SYS clock is slower (a minimum of 200 ns is required).

Note: Only one sample signal exists to perform sampling while entering DPD mode. Thus all interfaces are sampled at the same time. The sequence of multiple DPD groups requires keeping all interfaces (already in DPD mode) in idle.

DPD Request

Offset: 0x1b8 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000x000000000xx0000000)

Bit	Reset	Description
31:30	0x0	CODE: Code of operation for all set bits 0 = IDLE 1 = DPD_OFF 2 = DPD_ON
29	0x0	USB_IC: Defunct. Puts usb_ic_pad in/out of deep power down mode 0 = OFF 1 = ON
28	0x0	HDMI: Puts hdmi_pad in/out of Deep Power Down mode 0 = OFF 1 = ON
27	0x0	DDR_DATA: Defunct. 0 = OFF 1 = ON
26	0x0	DISC_ADDR_CMD: Defunct 0 = OFF 1 = ON
25	0x0	DDR_ADDR_CMD: Defunct 0 = OFF 1 = ON
24	0x0	POP_ADDR_CMD: Defunct 0 = OFF 1 = ON
23	0x0	POP_CLK: Defunct 0 = OFF 1 = ON
22	0x0	COMP: puts xm0_comp_pd_pad, xm0_comp_pu_pad in/out of deep power down mode
21	0x0	DISC_VTTGEN: Defunct. 0 = OFF 1 = ON
20	0x0	POP_VTTGEN: Defunct. 0 = OFF 1 = ON
19	0x0	HSIC: puts HSIC rail in/out of Deep Power Down mode 0 = OFF 1 = ON
17	0x0	AUDIO: Puts audio rail in/out of Deep Power Down mode 0 = OFF 1 = ON
16	0x0	VI: Defunct. 0 = OFF 1 = ON
15	0x0	BB: Puts the BB rail in/out of Deep Power Down mode 0 = OFF 1 = ON
14	0x0	UART: Puts the UART rail in/out of Deep Power Down mode 0 = OFF 1 = ON
13	0x0	NAND: Puts the NAND rail in/out of Deep Power Down mode 0 = OFF 1 = ON
12	0x0	USB_BIAS: Puts usb_bias in/out of Deep Power Down mode

Bit	Reset	Description
		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
9	0x0	USB0: Puts USB0 in/out of Deep Power Down mode 0 = OFF 1 = ON
6	0x0	PEX_CLK2: PEX clk2 pad-DPD control 0 = OFF 1 = ON
5	0x0	PEX_CLK1: PEX clk1 pad- DPD control 0 = OFF 1 = ON
4	0x0	PEX_BIAS:PEX bias pad- DPD control 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

9.2.112 APBDEV_PMC_IO_DPD_STATUS_0

Reflects the DPD status of each I/O rail.

DPD Status

Offset: 0x1bc │ Read/Write: R/W │ Reset: 0x0000000X (0bxx00000000000x000000000xx000xxxx)

Bit	R/W	Reset	Description
29	RW	0x0	USB_IC: Defunct. usb_ic_pad in/out of Deep Power Down mode 0 = OFF 1 = ON
28	RW	0x0	HDMI: hdmi_pad in Deep Power Down mode 0 = OFF 1 = ON
27	RW	0x0	DDR_DATA: Defunct. 0 = OFF 1 = ON
26	RW	0x0	DISC_ADDR_CMD: Defunct 0 = OFF 1 = ON
25	RW	0x0	DDR_ADDR_CMD: Defunct

Bit	R/W	Reset	Description
			0 = OFF 1 = ON
24	RW	0x0	POP_ADDR_CMD: Defunct 0 = OFF 1 = ON
23	RW	0x0	POP_CLK: Defunct 0 = OFF 1 = ON
22	RW	0x0	COMP: Puts xm0_comp_pd_pad, xm0_comp_pu_pad in Deep Power Down mode
21	RW	0x0	DISC_VTTGEN: Defunct 0 = OFF 1 = ON
20	RW	0x0	POP_VTTGEN: Defunct 0 = OFF 1 = ON
19	RW	0x0	HSIC: Puts HSIC rail in Deep Power Down mode 0 = OFF 1 = ON
17	RW	0x0	AUDIO: Puts audio rail in Deep Power Down mode 0 = OFF 1 = ON
16	RW	0x0	VI: Defunct. Puts VI rail in Deep Power Down mode 0 = OFF 1 = ON
15	RW	0x0	BB: BB rail in Deep Power Down mode 0 = OFF 1 = ON
14	RW	0x0	UART: UART rail in Deep Power Down mode 0 = OFF 1 = ON
13	RW	0x0	NAND: NAND rail in Deep Power Down mode 0 = OFF 1 = ON
12	RW	0x0	USB_BIAS: usb_bias in Deep Power Down mode 0 = OFF 1 = ON
11	RW	0x0	USB2: USB2 in Deep Power Down mode 0 = OFF 1 = ON
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
6	RW	0x0	PEX_CLK2:PEX clk2 pad-DPD control 0 = OFF 1 = ON
5	RW	0x0	PEX_CLK1:PEX clk1 pad-DPD control 0 = OFF 1 = ON

Bit	R/W	Reset	Description
4	RW	0x0	PEX_BIAS:PEX bias pad-DPD control 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

9.2.113 APBDEV_PMC_IO_DPD2_REQ_0

Second set of DPD requests due to additional rails.

Offset: 0x1c0 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:30	0x0	CODE: code of operation for all set bits 0 = IDLE 1 = DPD_OFF 2 = DPD_ON
29:27	0x0	NEW_RAILS: Reserved for future rails
26	0x0	SYS_DDC: for direcct DC pads 0 = OFF 1 = ON
25	0x0	LVDS: LVDS is in DPD by default 0 = OFF 1 = ON
24	0x0	DDR1_DATA: Defunct 0 = OFF 1 = ON
23	0x0	DDR1_CLK: Defunct 0 = OFF 1 = ON
22	0x0	DDR1_ADDR2_CMD: Defunct 0 = OFF 1 = ON
21	0x0	DDR1_ADDR1_CMD: Defunct 0 = OFF 1 = ON
20	0x0	DDR1_ADDR0_CMD: Defunct 0 = OFF 1 = ON
19	0x0	DDR0_CLK: Defunct 0 = OFF 1 = ON
18	0x0	DDR0_ADDR2_CMD: Defunct 0 = OFF

Bit	Reset	Description
		1 = ON
17	0x0	DDR0_ADDR1_CMD: Defunct 0 = OFF 1 = ON
16	0x0	DDR0_ADDR0_CMD: Defunct 0 = OFF 1 = ON
15	0x0	DISC_VTTGEN4: Defunct 0 = OFF 1 = ON
14	0x0	DISC_VTTGEN3: Defunct 0 = OFF 1 = ON
13	0x0	DISC_VTTGEN2: Defunct 0 = OFF 1 = ON
12	0x0	CSIE: Puts CSIE in/out of Deep Power Down mode 0 = OFF 1 = ON
11	0x0	CSID: Defunct 0 = OFF 1 = ON
10	0x0	CSIC: Defunct 0 = OFF 1 = ON
9	0x0	DSID: Puts DSID in/out of Deep Power Down mode 0 = OFF 1 = ON
8	0x0	DSIC: Puts DSIC in/out of Deep Power Down mode 0 = OFF 1 = ON
7	0x0	DSIB: Puts DSIB in/out of Deep Power Down mode 0 = OFF 1 = ON
6	0x0	HV: Puts HV rail in/out of Deep Power Down mode 0 = OFF 1 = ON
5	0x0	RES_RAIL: 0 = OFF 1 = ON
4	0x0	CAM: Puts the camera in/out of Deep Power Down mode 0 = OFF 1 = ON
3	0x0	SDMMC4: Puts SDMMC4 in/out of Deep Power Down mode 0 = OFF 1 = ON
2	0x0	SDMMC3:puts SDMMC3 in/out of Deep Power Down mode 0 = OFF 1 = ON
1	0x0	SDMMC1:puts SDMMC1 in/out of Deep Power Down mode 0 = OFF 1 = ON
0	0x0	PEX_CNTRL: Puts pex_cntrl in/out of Deep Power Down mode. 0 = OFF

Bit	Reset	Description
		1 = ON

9.2.114 APBDEV_PMC_IO_DPD2_STATUS_0

DPD2 Status

Offset: 0x1c4 | Read/Write: R/W | Reset: 0x02000000 (0bxx00001000000000000000000000000)

Bit	Reset	Description
29:27	0x0	NEW_RAILS: Reserved for future rails
26	0x0	SYS_DDC: For direct DC pads 0 = OFF 1 = ON
25	0x1	LVDS: LVDS is in DPD by default 0 = OFF 1 = ON
24	0x0	DDR1_DATA: Defunct 0 = OFF 1 = ON
23	0x0	DDR1_CLK: XM1 _MCLK: Defunct 0 = OFF 1 = ON
22	0x0	DDR1_ADDR2_CMD: Defunct 0 = OFF 1 = ON
21	0x0	DDR1_ADDR1_CMD: Defunct 0 = OFF 1 = ON
20	0x0	DDR1_ADDR0_CMD: Defunct 0 = OFF 1 = ON
19	0x0	DDR0_CLK: Defunct 0 = OFF 1 = ON
18	0x0	DDR0_ADDR2_CMD: Defunct 0 = OFF 1 = ON
17	0x0	DDR0_ADDR1_CMD: Defunct 0 = OFF 1 = ON
16	0x0	DDR0_ADDR0_CMD: Defunct 0 = OFF 1 = ON
15	0x0	DISC_VTTGEN4: Defunct 0 = OFF 1 = ON
14	0x0	DISC_VTTGEN3: Defunct 0 = OFF 1 = ON
13	0x0	DISC_VTTGEN2: Defunct 0 = OFF 1 = ON
12	0x0	CSIE:CSIE in/out of Deep Power Down mode 0 = OFF

Bit	Reset	Description
		1 = ON
11	0x0	CSID:CSID in/out of Deep Power Down mode 0 = OFF 1 = ON
10	0x0	CSIC:CSIC in/out of Deep Power Down mode 0 = OFF 1 = ON
9	0x0	DSID:DSID in/out of Deep Power Down mode 0 = OFF 1 = ON
8	0x0	DSIC:DSIC in/out of Deep Power Down mode 0 = OFF 1 = ON
7	0x0	DSIB: DSI in Deep Power Down mode 0 = OFF 1 = ON
6	0x0	HV: HV rail in/out of Deep Power Down mode 0 = OFF 1 = ON
5	0x0	RES_RAIL: reserved 0 = OFF 1 = ON
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 DPD status 0 = OFF 1 = ON

9.2.115 APBDEV_PMC_SEL_DPD_TIM_0

This timer guarantees proper timing spacing in hardware between the sel_dpd and e_dpd signals issued to pads.

A minimum of 200 ns is required, timer in APB/SYS clock.

Offset: 0x1c8 │ Read/Write: R/W │ Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111)

Bit	Reset	Description
6:0	0x7f	SEL_DPD_TIM: Timer which separates e_dpd deassertion time from sel_dpd deassertion time in apb_clk units.

9.2.116 APBDEV_PMC_VDDP_SEL_0

Power set for new DDR pads. Safe value is 11.

Offset: 0x1cc │ Read/Write: R/W │ Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx11)

Bit	Reset	Description
1:0	0x3	DATA: VDDP sel bits to DDR pads

9.2.117 APBDEV_PMC_DDR_CFG_0

Package Type for CAR/PMC Control

Offset: 0x1d0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxx000000000000000000)

Bit	Reset	Description
17	0x0	XM0_MCKE_B1_TRI: Tristate control for xm2_mcke_b_1_pad 0 = DISABLE 1 = ENABLE
16	0x0	XM0_MCKE_B0_TRI: Tristate control for xm2_mcke_b_0_pad 0 = DISABLE 1 = ENABLE
15	0x0	XM0_MCKE1_TRI: Tristate control for xm2_mcke_1_pad 0 = DISABLE 1 = ENABLE
14	0x0	XM0_MCKE0_TRI: Tristate control for xm2_mcke_0_pad 0 = DISABLE 1 = ENABLE
13	0x0	XM0_RESET_DPDIO_0: DPD output control for RESET pad 0 = DISABLE 1 = ENABLE
12	0x0	XM0_RESET_TRI: Tristate control for reset pad 0 = DISABLE 1 = ENABLE
11	0x0	XM1_MCLK_TRI: Defunct 0 = DISABLE 1 = ENABLE
10	0x0	XM0_MCLK_TRI: Defunct 0 = DISABLE 1 = ENABLE
9	0x0	XM1_TRI: Defunct 0 = DISABLE 1 = ENABLE
8	0x0	XM0_TRI: Defunct 0 = DISABLE 1 = ENABLE
7:4	0x0	RESET_SWIZZLE: Defunct
3	0x0	DDR_SPARE: Defunct
2	0x0	CHAN_SWIZZLE: Defunct 0 = DISABLE 1 = ENABLE
1	0x0	IF: Defunct 0 = LPDDR2 1 = DDR3
0	0x0	PKG: Defunct 0 = DISC 1 = POP

9.2.118 APBDEV_PMC_PLLM_WB0_OVERRIDE_FREQ_0

PLL WB Override Registers to Accelerate Warm Boot Time

Offset: 0x1dc │ Read/Write: R/W │ Reset: 0x0000010c (0bxxxxxxxxxxxxxxxx0000000100001100)

Bit	Reset	Description
15:8	0x1	PLLM_DIVN: PLL feedback divider.
7:0	0xc	PLLM_DIVM: PLL input divider.

9.2.119 APBDEV_PMC_TEST_PWRGATE_0

Force Test Power Gate Override Off/On

Offset: 0x1e0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	RESET_DEBUG: used for debug, assertion of reset will be disabled, if this bit is set 0 = OFF 1 = ON
3	0x0	DPD_ENABLE_DEBUG: used for debug, assertion of dpd_enable will be disabled if this bit is set 0 = OFF 1 = ON
2	0x0	MAIN_CLAMP_DEBUG: used for debug, assertion of main clamp will be disabled if this bit is set 0 = OFF 1 = ON
1:0	0x0	OP: FORCE_ON - force power gated partition to be powered, FORCE_OFF - force power gating partition to be powered down 0 = NONE 1 = FORCE_ON 2 = FORCE_OFF

9.2.120 APBDEV_PMC_PWRGATE_TIMER_MULT_0

The time for each rail set by PWRGATE_TIMER_OFF will be multiplied by MULT for power-gating up/down any power-gated region except CPU. In the CPU power-gated case, MULT_CPU will be used with PWRGATE_TIMER_CE* registers value. All timers are in sys_clk units.

Offset: 0x1e4 │ Read/Write: R/W │ Reset: 0x0000001b (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx011011)

Bit	Reset	Description
5:3	0x3	MULT_CPU: 0 = ONE 1 = TWO 2 = FOUR 3 = EIGHT 4 = SIXTEEN
2:0	0x3	MULT: 0 = ONE 1 = TWO 2 = FOUR 3 = EIGHT 4 = SIXTEEN

9.2.121 APBDEV_PMC_DSI_SEL_DPD_0

Register to control sel_dpd for the DSI pad. Allows driving LP0 value on the DSI pad beyond LP0 exit. This enables the DSI to be programmed properly at LP0 exit while the LP0 value is still driven by the brick pad.

Offset: 0x1e8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	0x0	SET_DSID: 0 = OFF 1 = ON
2	0x0	SET_DSIC: 0 = OFF 1 = ON
1	0x0	SET_DSIB: 0 = OFF 1 = ON
0	0x0	SET_DSIA: 0 = OFF 1 = ON

9.2.122 APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0

PMC trigger events for the UTMIP ports as well as the UHSIC port. Sleep walk clearing returns the walk pointer to 00, the first entry in a USB line walk. This is the only way to return a pointer to 0.

Pad configuration captures a set of FLLS parameters prior to resting the port so their value can be retained in deep sleep.

The trigger should only happen on the fields that are set to TRIG (1) when writing. The returned read value should always be all NULL fields.

The FORCE_WALK bits trigger a SLEEP.

Any value read back should be 0.

Triggers for USB Ports

Offset: 0x1ec │ Read/Write: R/W │ Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15	X	UHSIC_CLR_WAKE_ALARM_P0: Clear wake event for UHSIC port 0 0 = NULL 1 = TRIG
14	X	UTMIP_CLR_WAKE_ALARM_P2: Clear wake event for UTMIP port 2 0 = NULL 1 = TRIG
13	X	UTMIP_CLR_WAKE_ALARM_P1: Clear wake event for UTMIP port 1 0 = NULL 1 = TRIG
12	X	UTMIP_CLR_WAKE_ALARM_P0: Clear wake event for UTMIP port 0 0 = NULL 1 = TRIG
11	X	UHSIC_FORCE_WALK_P0: Force pointer walk for UHSIC port 0 0 = NULL 1 = TRIG
10	X	UTMIP_FORCE_WALK_P2: Force pointer walk for UTMIP port 2 0 = NULL 1 = TRIG

Bit	Reset	Description
9	X	UTMIP_FORCE_WALK_P1: Force pointer walk for UTMIP port 1 0 = NULL 1 = TRIG
8	X	UTMIP_FORCE_WALK_P0: Force pointer walk for UTMIP port 0 0 = NULL 1 = TRIG
7	X	UHSIC_RESERVED_P0: Reserved for UHSIC port 0 0 = NULL 1 = TRIG
6	X	UTMIP_CAP_CFG_P2: Capture pad configuration for UTMIP port 2 0 = NULL 1 = TRIG
5	X	UTMIP_CAP_CFG_P1: Capture pad configuration for UTMIP port 1 0 = NULL 1 = TRIG
4	X	UTMIP_CAP_CFG_P0: Capture pad configuration for UTMIP port 0 0 = NULL 1 = TRIG
3	X	UHSIC_CLR_WALK_PTR_P0: Clear sleep walk pointer for UHSIC port 0 0 = NULL 1 = TRIG
2	X	UTMIP_CLR_WALK_PTR_P2: Clear sleep walk pointer for UTMIP port 2 0 = NULL 1 = TRIG
1	X	UTMIP_CLR_WALK_PTR_P1: Clear sleep walk pointer for UTMIP port 1 0 = NULL 1 = TRIG
0	X	UTMIP_CLR_WALK_PTR_P0: Clear sleep walk pointer for UTMIP port 0 0 = NULL 1 = TRIG

9.2.123 APBDEV_PMC_UTMIP_UHSIC_SAVED_STATE_0

Save some critical information about USB port prior to entering DPD in a suspend state.

SPEED:

- HS - Suspend DPD was entered from a high speed mode, restore high speed at DPD exit
- FS - Suspend DPD was entered from a full speed mode, restore full speed at DPD exit
- LS - Suspend DPD was entered from a low speed mode, restore low speed at DPD exit
- RST - Reset Port, Hardware reset or DPD was not entered suspended bus, Reset and Re-enumerate port

SCRATCH -- save other critical information about the port, software choice.

Reset value should read 0x0F0F0F0F, reset should occur on Cold Hardware Reset only

Saved DPD State for all UTMIP and UHSIC Ports

Offset: 0x1f0 │ Read/Write: R/W │ Reset: 0x0f0f0f0f (0b00001111000011110000111100001111)

Bit	Reset	Description
31	0x0	UHSIC_WAKE_EX_P0: Wake up on anything except a Particular Line Value 0 = OFF 1 = ON
30	0x0	UHSIC_IGNORE_MASTER_CFG_P0

Bit	Reset	Description
29:25	0x7	UHSIC_SCRATCH_P0
24	0x1	UHSIC_MODE_P0: UHSIC Speed prior to DPD 0 = HS 1 = RST
23	0x0	UTMIP_WAKE_EX_P2: Wake up on anything except a Particular Line Value 0 = OFF 1 = ON
22	0x0	UTMIP_IGNORE_MASTER_CFG_P2
21:18	0x3	UTMIP_SCRATCH_P2
17:16	0x3	UTMIP_SPEED_P2: UTMIP Speed prior to DPD 0 = HS 1 = FS 2 = LS 3 = RST
15	0x0	UTMIP_WAKE_EX_P1: Wake up on anything except a Particular Line Value 0 = OFF 1 = ON
14	0x0	UTMIP_IGNORE_MASTER_CFG_P1
13:10	0x3	UTMIP_SCRATCH_P1
9:8	0x3	UTMIP_SPEED_P1: UTMIP Speed prior to DPD 0 = HS 1 = FS 2 = LS 3 = RST
7	0x0	UTMIP_WAKE_EX_P0: Wake up on anything except a Particular Line Value 0 = OFF 1 = ON
6	0x0	UTMIP_IGNORE_MASTER_CFG_P0
5:2	0x3	UTMIP_SCRATCH_P0
1:0	0x3	UTMIP_SPEED_P0: UTMIP Speed prior to DPD 0 = HS 1 = FS 2 = LS 3 = RST

9.2.124 APBDEV_PMC_UTMIP_PAD_CFG_0

Set of static configuration values for USB I/O pads under DPD mode. These were configuration values captured at a time prior to entering DPD. These values are read only. Registers must take into account DFT.

I/O Pad Config for Port 0

Offset: 0x1f4 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29	X	LSBIAS_SEL_P2: I/O pad LSBIAS_SEL for UTMIP P2
28	X	LO_SPD_P2: I/O pad LO_SPD for UTMIP P2
27:26	X	SPARE_P2: I/O pad SPARE1..0 for UTMIP P2
25:24	X	FS_SLEW_P2: I/O pad FS_SLEW1..0 for UTMIP P2
23:22	X	LS_FSLEW_P2: I/O pad LS_FSLEW1..0 for UTMIP P2
21:20	X	LS_RSLEW_P2: I/O pad LS_RSLEW1..0 for UTMIP P2

Bit	Reset	Description
19	X	LSBIAS_SEL_P1: I/O pad LSBIAS_SEL for UTMIP P1
18	X	LO_SPD_P1: I/O pad LO_SPD for UTMIP P1
17:16	X	SPARE_P1: I/O pad SPARE1..0 for UTMIP P1
15:14	X	FS_SLEW_P1: I/O pad FS_SLEW1..0 for UTMIP P1
13:12	X	LS_FSLEW_P1: I/O pad LS_FSLEW1..0 for UTMIP P1
11:10	X	LS_RSLEW_P1: I/O pad LS_RSLEW1..0 for UTMIP P1
9	X	LSBIAS_SEL_P0: I/O pad LSBIAS_SEL for UTMIP P0
8	X	LO_SPD_P0: I/O pad LO_SPD for UTMIP P0
7:6	X	SPARE_P0: I/O pad SPARE1..0 for UTMIP P0
5:4	X	FS_SLEW_P0: I/O pad FS_SLEW1..0 for UTMIP P0
3:2	X	LS_FSLEW_P0: I/O pad LS_FSLEW1..0 for UTMIP P0
1:0	X	LS_RSLEW_P0: I/O pad LS_RSLEW1..0 for UTMIP P0

9.2.125 APBDEV_PMC_UTMIP_TERM_PAD_CFG_0

Set of termination configuration values for USB I/O pads under DPD mode. These configuration values are programmed, not captured, at a time prior to entering DPD. Capturing them would be complex because it would require costly synchronizers as well as some logic. The range for RCTRL and TCTRL is 0 to 16.

These values need to be converted to a thermal encoding (only one 0 to 1 transition allowed in the word from MSB to LSB). For example:

0 - 0000_0000_0000_0000

1 - 0000_0000_0000_0001

2 - 0000_0000_0000_0011

...

15 - 0111_1111_1111_1111

16 - 1111_1111_1111_1111

I/O Termination Configuration for all UTMIP Ports

Offset: 0x1f8 │ Read/Write: R/W │ Reset: 0x00000108 (0bxxxxxxxxxxxxxxxxxxxxxxxx0100001000)

Bit	Reset	Description
9:5	0x8	TCTRL_VAL: HS termination calibration value, range 0 to 16
4:0	0x8	RCTRL_VAL: 1.5kOhm pull up calibration value, range 0 to 16

9.2.126 APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0

Register that configures the value of the line that could cause a wake-up event.

- ANY -- signifies any line change DM EDGE or DP EDGE for UTMIP.
- NONE -- signifies no wakeup possible.

One can use the two most significant bits (3:2) of the WAKE_VAL to 4x1 select whether:

- 00: Check both bits for value on two LSB (1:0).
- 01: Check bit DM (or strobe for HSIC) against bit 0

- 10: Check bit DP (or DATA for HSIC) against bit 1
- 11: Check for edge on DP against bit 1, edge on DM against bit 0

Sleep Walk Sequence Enables

Offset: 0x1fc │ Read/Write: R/W │ Reset: 0xc0c0c0c0 (0b11000000110000001100000011000000)

Bit	Reset	Description
31:28	0xc	UHSIC_WAKE_VAL_P0: Line Value Wake Up Condition on UHSIC P0 0 = SD00 1 = SD01 2 = SD10 3 = SD11 4 = S0 5 = S1 8 = D0 10 = D1 12 = NONE 13 = SEDGE 14 = DEDGE 15 = ANY
27:25	0x0	UHSIC_RESERVED_P0: Reserved for later use for UHSIC P0
24	0x0	UHSIC_MASTER_ENABLE_P0: Enable use of master pins on UHSIC P0
23:20	0xc	UTMIP_WAKE_VAL_P2: Line Value Wake Up Condition on UTMIP P2 0 = SE0 1 = FSK 2 = FSJ 3 = SE1 4 = DM0 5 = DM1 8 = DP0 10 = DP1 12 = NONE 13 = DMEDGE 14 = DPEDGE 15 = ANY
19	0x0	UTMIP_TCTRL_USE_PMC_P2: Use PMC Saved TCTRL on UTMIP P2
18	0x0	UTMIP_RCTRL_USE_PMC_P2: Use PMC Saved RCTRL on UTMIP P2
17	0x0	UTMIP_FSLS_USE_PMC_P2: Use PMC Saved Pad config on UTMIP P2
16	0x0	UTMIP_MASTER_ENABLE_P2: Enable use of master pins on UTMIP P2
15:12	0xc	UTMIP_WAKE_VAL_P1: Line Value Wake Up Condition on UTMIP P1 0 = SE0 1 = FSK 2 = FSJ 3 = SE1 4 = DM0 5 = DM1 8 = DP0 10 = DP1 12 = NONE 13 = DMEDGE 14 = DPEDGE 15 = ANY
11	0x0	UTMIP_TCTRL_USE_PMC_P1: Use PMC Saved TCTRL on UTMIP P1
10	0x0	UTMIP_RCTRL_USE_PMC_P1: Use PMC Saved RCTRL on UTMIP P1
9	0x0	UTMIP_FSLS_USE_PMC_P1: Use PMC Saved Pad config on UTMIP P1
8	0x0	UTMIP_MASTER_ENABLE_P1: Enable use of master pins on UTMIP P1
7:4	0xc	UTMIP_WAKE_VAL_P0: Line Value Wake Up Condition on UTMIP P0

Bit	Reset	Description
		0 = SE0 1 = FSK 2 = FSJ 3 = SE1 4 = DM0 5 = DM1 8 = DP0 10 = DP1 12 = NONE 13 = DMEDGE 14 = DPEDGE 15 = ANY
3	0x0	UTMIP_TCTRL_USE_PMC_P0: Use PMC Saved TCTRL on UTMIP P0
2	0x0	UTMIP_RCTRL_USE_PMC_P0: Use PMC Saved RCTRL on UTMIP P0
1	0x0	UTMIP_FSLS_USE_PMC_P0: Use PMC Saved Pad config on UTMIP P0
0	0x0	UTMIP_MASTER_ENABLE_P0: Enable use of master pins on UTMIP P0

9.2.127 APBDEV_PMC_UTMIP_UHSIC_SLEEPWALK_CFG_0

This register determines when sleep walking should take effect: upon a specific GPIO event, on any wake event, or on a line value change.

It is also possible to force a sleep walk; see register UTMIP_UHSIC_TRIGGERS to force the event.

Sleep Walk Sequence Enables

Offset: 0x200 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	UHSIC_LINEVAL_WALK_EN_P0: Perform Walk on USB line value wake up for UHSIC P0
30	0x0	UHSIC_WAKE_WALK_EN_P0: Perform Walk on any chip wake up event for UHSIC P0
29	0x0	UHSIC_GPIO_WALK_EN_P0: Perform Walk on associated GPIO event for UHSIC P0
28:24	0x0	UHSIC_DESIGNATED_GPIO_P0: GPIO Number associated with UHSIC P0
23	0x0	UTMIP_LINEVAL_WALK_EN_P2: Perform Walk on USB line value wake up for UTMIP P2
22	0x0	UTMIP_WAKE_WALK_EN_P2: Perform Walk on any chip wake up event for UTMIP P2
21	0x0	UTMIP_GPIO_WALK_EN_P2: Perform Walk on associated GPIO event for UTMIP P2
20:16	0x0	UTMIP_DESIGNATED_GPIO_P2: GPIO Number associated with UTMIP P2
15	0x0	UTMIP_LINEVAL_WALK_EN_P1: Perform Walk on USB line value wake up for UTMIP P1
14	0x0	UTMIP_WAKE_WALK_EN_P1: Perform Walk on any chip wake up event for UTMIP P1
13	0x0	UTMIP_GPIO_WALK_EN_P1: Perform Walk on associated GPIO event for UTMIP P1
12:8	0x0	UTMIP_DESIGNATED_GPIO_P1: GPIO Number associated with UTMIP P1
7	0x0	UTMIP_LINEVAL_WALK_EN_P0: Perform Walk on USB line value wake up for UTMIP P0
6	0x0	UTMIP_WAKE_WALK_EN_P0: Perform Walk on any chip wake up event for UTMIP P0
5	0x0	UTMIP_GPIO_WALK_EN_P0: Perform Walk on associated GPIO event for UTMIP P0
4:0	0x0	UTMIP_DESIGNATED_GPIO_P0: GPIO Number associated with UTMIP P0

9.2.128 APBDEV_PMC_UTMIP_SLEEPWALK_P0_0

These registers hold the sequence of control values to the UTMIP pads on the four consecutive cycles of a walk. The pad pins that are controlled are:

- MASTER_USBOP_RPD
- MASTER_USBON_RPD
- MASTER_USBOP_RPU
- MASTER_USBON_RPU
- MASTER_AP
- MASTER_AN
- MASTER_HIGHZ

For the walk to take effect at the pad, the MASTER_ENABLE pin must be set high in the config register. Otherwise the pad will ignore the values.

If no walk is enabled or forced, then the walk pointer remains stuck on phase A. The walk pointer should use a 2 bit Gray code so that Phase A is 00, Phase B is 01, Phase C is 11, and Phase D is 10. Once Phase D is reached, only a reset of the phase pointer can bring it back to Phase A.

Four phases should be sufficient to handle most wake-up events.

Signaling Sequence for UTMIP Port 0 Wakeup

Offset: 0x204 │ Read/Write: R/W │ Reset: 0x23232363 (0b00100011001000110010001101100011)

Bit	Reset	Description
31	0x0	RESERVED_D: Phase D Unused Bit
30	0x0	HIGHZ_D: Phase D Enable Single Ended Drivers
29	0x1	AN_D: Phase D Drive Single Ended Value on D- line
28	0x0	AP_D: Phase D Drive Single Ended Value on D+ line
27	0x0	USBON_RPU_D: Phase D 1.5kOhm Pull up on D- Line
26	0x0	USBOP_RPU_D: Phase D 1.5kOhm Pull Up on D+ Line
25	0x1	USBON_RPD_D: Phase D 15kOhm Pull Down on D- Line
24	0x1	USBOP_RPD_D: Phase D 15kOhm Pull Down on D+ Line
23	0x0	RESERVED_C: Phase C Unused Bit
22	0x0	HIGHZ_C: Phase C Enable Single Ended Drivers
21	0x1	AN_C: Phase C Drive Single Ended Value on D- line
20	0x0	AP_C: Phase C Drive Single Ended Value on D+ line
19	0x0	USBON_RPU_C: Phase C 1.5kOhm Pull up on D- Line
18	0x0	USBOP_RPU_C: Phase C 1.5kOhm Pull Up on D+ Line
17	0x1	USBON_RPD_C: Phase C 15kOhm Pull Down on D- Line
16	0x1	USBOP_RPD_C: Phase C 15kOhm Pull Down on D+ Line
15	0x0	RESERVED_B: Phase B Unused Bit
14	0x0	HIGHZ_B: Phase B Enable Single Ended Drivers
13	0x1	AN_B: Phase B Drive Single Ended Value on D- line
12	0x0	AP_B: Phase B Drive Single Ended Value on D+ line

Bit	Reset	Description
11	0x0	USBON_RPU_B: Phase B 1.5kOhm Pull up on D- Line
10	0x0	USBOP_RPU_B: Phase B 1.5kOhm Pull Up on D+ Line
9	0x1	USBON_RPD_B: Phase B 15kOhm Pull Down on D- Line
8	0x1	USBOP_RPD_B: Phase B 15kOhm Pull Down on D+ Line
7	0x0	RESERVED_A: Phase A Unused Bit
6	0x1	HIGHZ_A: Phase A Enable Single Ended Drivers
5	0x1	AN_A: Phase A Drive Single Ended Value on D- line
4	0x0	AP_A: Phase A Drive Single Ended Value on D+ line
3	0x0	USBON_RPU_A: Phase A 1.5kOhm Pull up on D- Line
2	0x0	USBOP_RPU_A: Phase A 1.5kOhm Pull Up on D+ Line
1	0x1	USBON_RPD_A: Phase A 15kOhm Pull Down on D- Line
0	0x1	USBOP_RPD_A: Phase A 15kOhm Pull Down on D+ Line

9.2.129 APBDEV_PMC_UTMIP_SLEEPWALK_P1_0

Signaling Sequence for UTMIP Port 1 Wakeup

Offset: 0x208 │ Read/Write: R/W │ Reset: 0x23232363 (0b00100011001000110010001101100011)

Bit	Reset	Description
31	0x0	RESERVED_D: Phase D Unused Bit
30	0x0	HIGHZ_D: Phase D Enable Single Ended Drivers
29	0x1	AN_D: Phase D Drive Single Ended Value on D- line
28	0x0	AP_D: Phase D Drive Single Ended Value on D+ line
27	0x0	USBON_RPU_D: Phase D 1.5kOhm Pull up on D- Line
26	0x0	USBOP_RPU_D: Phase D 1.5kOhm Pull Up on D+ Line
25	0x1	USBON_RPD_D: Phase D 15kOhm Pull Down on D- Line
24	0x1	USBOP_RPD_D: Phase D 15kOhm Pull Down on D+ Line
23	0x0	RESERVED_C: Phase C Unused Bit
22	0x0	HIGHZ_C: Phase C Enable Single Ended Drivers
21	0x1	AN_C: Phase C Drive Single Ended Value on D- line
20	0x0	AP_C: Phase C Drive Single Ended Value on D+ line
19	0x0	USBON_RPU_C: Phase C 1.5kOhm Pull up on D- Line
18	0x0	USBOP_RPU_C: Phase C 1.5kOhm Pull Up on D+ Line
17	0x1	USBON_RPD_C: Phase C 15kOhm Pull Down on D- Line
16	0x1	USBOP_RPD_C: Phase C 15kOhm Pull Down on D+ Line
15	0x0	RESERVED_B: Phase B Unused Bit
14	0x0	HIGHZ_B: Phase B Enable Single Ended Drivers
13	0x1	AN_B: Phase B Drive Single Ended Value on D- line
12	0x0	AP_B: Phase B Drive Single Ended Value on D+ line
11	0x0	USBON_RPU_B: Phase B 1.5kOhm Pull up on D- Line

Bit	Reset	Description
10	0x0	USBOP_RPU_B: Phase B 1.5kOhm Pull Up on D+ Line
9	0x1	USBON_RPD_B: Phase B 15kOhm Pull Down on D- Line
8	0x1	USBOP_RPD_B: Phase B 15kOhm Pull Down on D+ Line
7	0x0	RESERVED_A: Phase A Unused Bit
6	0x1	HIGHZ_A: Phase A Enable Single Ended Drivers
5	0x1	AN_A: Phase A Drive Single Ended Value on D- line
4	0x0	AP_A: Phase A Drive Single Ended Value on D+ line
3	0x0	USBON_RPU_A: Phase A 1.5kOhm Pull up on D- Line
2	0x0	USBOP_RPU_A: Phase A 1.5kOhm Pull Up on D+ Line
1	0x1	USBON_RPD_A: Phase A 15kOhm Pull Down on D- Line
0	0x1	USBOP_RPD_A: Phase A 15kOhm Pull Down on D+ Line

9.2.130 APBDEV_PMC_UTMIP_SLEEPWALK_P2_0

Signaling Sequence for UTMIP Port 2 Wakeup

Offset: 0x20c │ Read/Write: R/W │ Reset: 0x23232363 (0b00100011001000110010001101100011)

Bit	Reset	Description
31	0x0	RESERVED_D: Phase D Unused Bit
30	0x0	HIGHZ_D: Phase D Enable Single Ended Drivers, active low
29	0x1	AN_D: Phase D Drive Single Ended Value on D- line
28	0x0	AP_D: Phase D Drive Single Ended Value on D+ line
27	0x0	USBON_RPU_D: Phase D 1.5kOhm Pull up on D- Line
26	0x0	USBOP_RPU_D: Phase D 1.5kOhm Pull Up on D+ Line
25	0x1	USBON_RPD_D: Phase D 15kOhm Pull Down on D- Line
24	0x1	USBOP_RPD_D: Phase D 15kOhm Pull Down on D+ Line
23	0x0	RESERVED_C: Phase C Unused Bit
22	0x0	HIGHZ_C: Phase C Enable Single Ended Drivers, active low
21	0x1	AN_C: Phase C Drive Single Ended Value on D- line
20	0x0	AP_C: Phase C Drive Single Ended Value on D+ line
19	0x0	USBON_RPU_C: Phase C 1.5kOhm Pull up on D- Line
18	0x0	USBOP_RPU_C: Phase C 1.5kOhm Pull Up on D+ Line
17	0x1	USBON_RPD_C: Phase C 15kOhm Pull Down on D- Line
16	0x1	USBOP_RPD_C: Phase C 15kOhm Pull Down on D+ Line
15	0x0	RESERVED_B: Phase B Unused Bit
14	0x0	HIGHZ_B: Phase B Enable Single Ended Drivers, active low
13	0x1	AN_B: Phase B Drive Single Ended Value on D- line
12	0x0	AP_B: Phase B Drive Single Ended Value on D+ line
11	0x0	USBON_RPU_B: Phase B 1.5kOhm Pull up on D- Line
10	0x0	USBOP_RPU_B: Phase B 1.5kOhm Pull Up on D+ Line

Bit	Reset	Description
9	0x1	USBON_RPD_B: Phase B 15kOhm Pull Down on D- Line
8	0x1	USBOP_RPD_B: Phase B 15kOhm Pull Down on D+ Line
7	0x0	RESERVED_A: Phase A Unused Bit
6	0x1	HIGHZ_A: Phase A Enable Single Ended Drivers, active low
5	0x1	AN_A: Phase A Drive Single Ended Value on D- line
4	0x0	AP_A: Phase A Drive Single Ended Value on D+ line
3	0x0	USBON_RPU_A: Phase A 1.5kOhm Pull up on D- Line
2	0x0	USBOP_RPU_A: Phase A 1.5kOhm Pull Up on D+ Line
1	0x1	USBON_RPD_A: Phase A 15kOhm Pull Down on D- Line
0	0x1	USBOP_RPD_A: Phase A 15kOhm Pull Down on D+ Line

9.2.131 APBDEV_PMC_UHSIC_SLEEPWALK_P0_0

These registers hold the sequence of control values to the UTMIP pads on the four consecutive cycles of a walk. The pad pins that are controlled are:

- STROBE_RPD
- DATA_RPD
- STROBE_RPU
- DATA_RPU

At this time there are no STROBE, DATA ports that should be driven by the host. So these outputs from the module should be left unconnected until a time that departure may be implemented.

Signaling Sequence for UHSIC Port 0 Wakeup

Offset: 0x210 │ Read/Write: R/W │ Reset: 0x06060606 (0b00000110000001100000011000000110)

Bit	Reset	Description
31:28	0x0	RESERVED_D: Phase D Unused Bit
27	0x0	UHSIC_DATA_RPU_D: Phase D Pull up on DATA Line
26	0x1	UHSIC_STROBE_RPU_D: Phase D Pull Up on STROBE Line
25	0x1	UHSIC_DATA_RPD_D: Phase D Pull Down on DATA Line
24	0x0	UHSIC_STROBE_RPD_D: Phase D Pull Down on STROBE Line
23:20	0x0	RESERVED_C: Phase C Unused Bit
19	0x0	UHSIC_DATA_RPU_C: Phase C Pull up on DATA Line
18	0x1	UHSIC_STROBE_RPU_C: Phase C Pull Up on STROBE Line
17	0x1	UHSIC_DATA_RPD_C: Phase C Pull Down on DATA Line
16	0x0	UHSIC_STROBE_RPD_C: Phase C Pull Down on STROBE Line
15:12	0x0	RESERVED_B: Phase B Unused Bit
11	0x0	UHSIC_DATA_RPU_B: Phase B Pull up on DATA Line
10	0x1	UHSIC_STROBE_RPU_B: Phase B Pull Up on STROBE Line
9	0x1	UHSIC_DATA_RPD_B: Phase B Pull Down on DATA Line
8	0x0	UHSIC_STROBE_RPD_B: Phase B Pull Down on STROBE Line

Bit	Reset	Description
7:4	0x0	RESERVED_A: Phase A Unused Bit
3	0x0	UHSIC_DATA_RPU_A: Phase A Pull up on DATA Line
2	0x1	UHSIC_STROBE_RPU_A: Phase A Pull Up on STROBE Line
1	0x1	UHSIC_DATA_RPD_A: Phase A Pull Down on DATA Line
0	0x0	UHSIC_STROBE_RPD_A: Phase A Pull Down on STROBE Line

9.2.132 APBDEV_PMC_UTMIP_UHSIC_STATUS_0

Read-only register that provides current walk pointer information as well as line value.

Status of UTMIP UHSIC Wake-up Circuitry

Offset: 0x214 │ Read/Write: RO │ Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19	X	UHSIC_WAKE_ALARM_P0: A wake event occurred on UHSIC port 0
18	X	UTMIP_WAKE_ALARM_P2: A wake event occurred on UTMIP port 2
17	X	UTMIP_WAKE_ALARM_P1: A wake event occurred on UTMIP port 1
16	X	UTMIP_WAKE_ALARM_P0: A wake event occurred on UTMIP port 0
15	X	DATA_VAL_P0: Value of DATA line detector for UHSIC port 0
14	X	STROBE_VAL_P0: Value of STROBE line detector for UHSIC port 0
13	X	USBON_VAL_P2: Value of D- line detector for UTMIP port 2
12	X	USBOP_VAL_P2: Value of D+ line detector for UTMIP port 2
11	X	USBON_VAL_P1: Value of D- line detector for UTMIP port 1
10	X	USBOP_VAL_P1: Value of D+ line detector for UTMIP port 1
9	X	USBON_VAL_P0: Value of D- line detector for UTMIP port 0
8	X	USBOP_VAL_P0: Value of D+ line detector for UTMIP port 0
7:6	X	UHSIC_WALK_PTR_P0: Walk pointer for UHSIC port 0
5:4	X	UTMIP_WALK_PTR_P2: Walk pointer for UTMIP port 2
3:2	X	UTMIP_WALK_PTR_P1: Walk pointer for UTMIP port 1
1:0	X	UTMIP_WALK_PTR_P0: Walk pointer for UMTIP port 0

9.2.133 APBDEV_PMC_UTMIP_UHSIC_FAKE_0

Instead of using the pad value for USBOP_VAL, USBON_VAL, STROBE_VAL, DATA_VAL, VBUS_WAKEUP, ID from the UTMIP and HSIC pads, force a 2x1 mux at the input of the PMC pad macro when fake values are enabled. This could the useful feature of putting a line at rest when needed. It also has important debug merits. This mux should be placed in front of the NP synchronizer for these signals (saved power on the synchronizer). This will have to be waived in our sync checks.

Fake the Line Value for the PMC Pad Macro

Offset: 0x218 │ Read/Write: R/W │ Reset: 0x01111111 (0bxxxx00010001000100010001000100010001)

Bit	Reset	Description
27	0x0	UTMIP_ID_FAKE_EN_P2: Enable the fake ID value for UTMIP P2

Bit	Reset	Description
26	0x0	UTMIP_ID_FAKE_VAL_P2: Fake ID value for UTMIP P2
25	0x0	UTMIP_VBUS_FAKE_EN_P2: Enable the fake VBUS WAKEUP value for UTMIP P2
24	0x1	UTMIP_VBUS_FAKE_VAL_P2: Fake VBUS WAKEUP value for UTMIP P2
23	0x0	UTMIP_ID_FAKE_EN_P1: Enable the fake ID value for UTMIP P1
22	0x0	UTMIP_ID_FAKE_VAL_P1: Fake ID value for UTMIP P1
21	0x0	UTMIP_VBUS_FAKE_EN_P1: Enable the fake VBUS WAKEUP value for UTMIP P1
20	0x1	UTMIP_VBUS_FAKE_VAL_P1: Fake VBUS WAKEUP value for UTMIP P1
19	0x0	UTMIP_ID_FAKE_EN_P0: Enable the fake ID value for UTMIP P0
18	0x0	UTMIP_ID_FAKE_VAL_P0: Fake ID value for UTMIP P0
17	0x0	UTMIP_VBUS_FAKE_EN_P0: Enable the fake VBUS WAKEUP value for UTMIP P0
16	0x1	UTMIP_VBUS_FAKE_VAL_P0: Fake VBUS WAKEUP value for UTMIP P0
15	0x0	UHSIC_FAKE_DATA_EN_P0: Enable the fake line value for DATA for the PMC pad macro for UHSIC P0
14	0x0	UHSIC_FAKE_STROBE_EN_P0: Enable the fake line value for STROBE for the PMC pad macro for UHSIC P0
13	0x0	UHSIC_FAKE_DATA_VAL_P0: Fake line value for DATA for the PMC pad macro for UHSIC P0
12	0x1	UHSIC_FAKE_STROBE_VAL_P0: Fake line value for STROBE for the PMC pad macro for UHSIC P0
11	0x0	UTMIP_FAKE_USBON_EN_P2: Enable the fake line value for D- for the PMC pad macro for UTMIP P2
10	0x0	UTMIP_FAKE_USBOP_EN_P2: Enable the fake line value for D+ for the PMC pad macro for UTMIP P2
9	0x0	UTMIP_FAKE_USBON_VAL_P2: Fake line value for D- for the PMC pad macro for UTMIP P2
8	0x1	UTMIP_FAKE_USBOP_VAL_P2: Fake line value for D+ for the PMC pad macro for UTMIP P2
7	0x0	UTMIP_FAKE_USBON_EN_P1: Enable the fake line value for D- for the PMC pad macro for UTMIP P1
6	0x0	UTMIP_FAKE_USBOP_EN_P1: Enable the fake line value for D+ for the PMC pad macro for UTMIP P1
5	0x0	UTMIP_FAKE_USBON_VAL_P1: Fake line value for D- for the PMC pad macro for UTMIP P1
4	0x1	UTMIP_FAKE_USBOP_VAL_P1: Fake line value for D+ for the PMC pad macro for UTMIP P1
3	0x0	UTMIP_FAKE_USBON_EN_P0: Enable the fake line value for D- for the PMC pad macro for UTMIP P0
2	0x0	UTMIP_FAKE_USBOP_EN_P0: Enable the fake line value for D+ for the PMC pad macro for UTMIP P0
1	0x0	UTMIP_FAKE_USBON_VAL_P0: Fake line value for D- for the PMC pad macro for UTMIP P0
0	0x1	UTMIP_FAKE_USBOP_VAL_P0: Fake line value for D+ for the PMC pad macro for UTMIP P0

9.2.134 APBDEV_PMC_BONDOUT_MIRROR3_0

Secure Scratch Register

Offset: 0x21c │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR3

9.2.135 APBDEV_PMC_BONDOUT_MIRROR4_0

Secure Scratch Register

Offset: 0x220 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR4

9.2.136 APBDEV_PMC_SECURE_SCRATCH6_0

Secure Scratch Register

Offset: 0x224 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH6

9.2.137 APBDEV_PMC_SECURE_SCRATCH7_0

Secure Scratch Register

Offset: 0x228 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH7

9.2.138 APBDEV_PMC_SCRATCH43_0

Scratch Register

Offset: 0x22c │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH43: General-purpose register storage. Currently used for power-gating progress status in RTAPI.

9.2.139 APBDEV_PMC_SCRATCH44_0

Scratch Register

Offset: 0x230 | Read/Write: R/W V Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH44: General-purpose register storage. Currently used for storing ARM SP in power-gating by RTAPI.

9.2.140 APBDEV_PMC_SCRATCH45_0

Scratch Register

Offset: 0x234 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH45: General-purpose register storage

9.2.141 APBDEV_PMC_SCRATCH46_0

Scratch Register

Offset: 0x238 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH46: General-purpose register storage

9.2.142 APBDEV_PMC_SCRATCH47_0

Scratch Register

Offset: 0x23c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH47: General-purpose register storage

9.2.143 APBDEV_PMC_SCRATCH48_0

Scratch Register

Offset: 0x240 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH48: General-purpose register storage

9.2.144 APBDEV_PMC_SCRATCH49_0

Scratch Register

Offset: 0x244 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH49: General-purpose register storage

9.2.145 APBDEV_PMC_SCRATCH50_0

Scratch Register

Offset: 0x248 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH50: General-purpose register storage

9.2.146 APBDEV_PMC_SCRATCH51_0

Scratch Register

Offset: 0x24c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH51: General-purpose register storage

9.2.147 APBDEV_PMC_SCRATCH52_0

Scratch Register

This register is valid in I2C mode. It is used for the WATCHDOG_RESET I2C command. In GPIO and SPI modes, this register is reserved.

Offset: 0x250 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	SCRATCH_PMU_A_0_HIWORD_RANGE
15:0	X	SCRATCH_PMU_A_0_LOWORD_RANGE

9.2.148 APBDEV_PMC_SCRATCH53_0

Scratch Register

This register is valid in I2C mode. It is used for the WATCHDOG_RESET I2C command. In GPIO and SPI modes, this register is reserved.

Offset: 0x254 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SCRATCH_PMU_B_0_RST_ENABLE_RANGE: Reset enable.
30	X	SCRATCH_PMU_B_0_CNTLR_TYPE_RANGE: This bit is reserved. Set it to 0.
29:27	X	SCRATCH_PMU_B_0_CNTLR_ID_RANGE: Controller ID. 0: I2C1 1: I2C2 2: I2C3 3: I2C4 4: I2C PMU
26:24	X	SCRATCH_PMU_B_0_PINMUX_RANGE. Pinmux Range. Refer to “Pinmux Support” in the Boot Process section for the encoding of this field.
23:16	X	SCRATCH_PMU_B_0_CHKSUM_RANGE: Checksum Range. Every byte in this register must add to 0. Refer to “Checksum Calculation” in the Boot Process section for the encoding of this field.

Bit	Reset	Description
15	X	SCRATCH_PMU_B_0_16BITOP_RANGE: 16-Bit Op.
14	X	SCRATCH_PMU_B_0_USE_GPIO_RANGE. Set this bit to 0.
13:7	X	Reserved. Set these bits to 0.
6:0	X	SCRATCH_PMU_B_0_I2CSLV1_RANGE: I2C Slave Address.

9.2.149 APBDEV_PMC_SCRATCH54_0

Scratch Register

This register is valid in I2C mode. It is used for the TSENSE_RESET I2C command. In GPIO and SPI modes, this register is reserved.

Offset: 0x258 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	SCRATCH_PMU_A_0_HIWORD_RANGE
15:0	X	SCRATCH_PMU_A_0_LOWORD_RANGE

9.2.150 APBDEV_PMC_SCRATCH55_0

Scratch Register

This register is valid in I2C mode. It is used for the TSENSE_RESET I2C command. In GPIO and SPI modes, this register is reserved.

Offset: 0x25c │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SCRATCH_PMU_B_0_RST_ENABLE_RANGE: Reset enable.
30	X	SCRATCH_PMU_B_0_CNTLR_TYPE_RANGE: This bit is reserved. Set it to 0.
29:27	X	SCRATCH_PMU_B_0_CNTLR_ID_RANGE: Controller ID. 0: I2C1 1: I2C2 2: I2C3 3: I2C4 4: I2C PMU
26:24	X	SCRATCH_PMU_B_0_PINMUX_RANGE. Pinmux Range. Refer to “Pinmux Support” in the Boot Process section for the encoding of this field.
23:16	X	SCRATCH_PMU_B_0_CHKSUM_RANGE: Checksum Range. Every byte in this register must add to 0. Refer to “Checksum Calculation” in the Boot Process section for the encoding of this field.
15	X	SCRATCH_PMU_B_0_16BITOP_RANGE: 16-Bit Op.
14	X	SCRATCH_PMU_B_0_USE_GPIO_RANGE. Set this bit to 0.
13:7	X	Reserved. Set these bits to 0.
6:0	X	SCRATCH_PMU_B_0_I2CSLV1_RANGE: I2C Slave Address.

9.2.151 APBDEV_PMC_SCRATCH0_ECO_0

Scratch Register

Offset: 0x260 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	CAR_USE_NEW_LP0_EXIT_SEQ: Reserved.
30:0	0x0	ECO0: Reserved.

9.2.152 APBDEV_PMC_POR_DPD_CTRL_0

Offset: 0x264 | Read/Write: R/W | Reset: 0x80000003 (0b1xxxxxxxxxxxxxxxxxxxxxxxxxxxxx11)

Bit	Reset	Description
31	0x1	MEM0_HOLD_CKE_LOW_OVR: Register control to pull CKE low. Using HOLD_CKE_LOW to guarantee that CKE stays low when DDR I/O rails are ramping up.
1	0x1	MEM0_ADDR1_CLK_SEL_DPD: sets sel_dpd to clock buffer.
0	0x1	MEM0_ADDR0_CLK_SEL_DPD: sets sel_dpd to clock buffer.

9.2.153 APBDEV_PMC_SCRATCH2_ECO_0

Scratch Register

Offset: 0x268 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	ECO2: Reserved.

9.2.154 APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0

Offset: 0x26c | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111)

Bit	Reset	Description
3	0x1	UHSIC_LINE_WAKEUP_EN_P0: enables latching line wakeup event on UHSIC p0
2	0x1	UTMIP_LINE_WAKEUP_EN_P2: enables latching line wakeup event on UTMIP p2
1	0x1	UTMIP_LINE_WAKEUP_EN_P1: enables latching line wakeup event on UTMIP p1
0	0x1	UTMIP_LINE_WAKEUP_EN_P0: enables latching line wakeup event on UTMIP p0

9.2.155 APBDEV_PMC_UTMIP_BIAS_MASTER_CNTRL_0

Offset: 0x270 | Read/Write: R/W | Reset: 0x0000000d (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1101)

Bit	Reset	Description
3	0x1	UTMIP_BIAS_MASTER_AWAKE_AND: When not PROGRAMMABLE and E_DPD=0, AND the MASTER_ENABLE of all IO ports for bias cell. If not AND or OR, force to 0.
2	0x1	UTMIP_BIAS_MASTER_AWAKE_OR: When not PROGRAMMABLE and E_DPD=0, OR the MASTER_ENABLE of all IO ports for bias cell
1	0x0	UTMIP_BIAS_MASTER_PROG_VAL: When PROGRAMMABLE, this is the value to program too
0	0x1	UTMIP_BIAS_MASTER_PROG_CTRL: MASTER_ENABLE pin uses a programmable value on BIAS pad, at all times

9.2.156 APBDEV_PMC_UTMIP_MASTER_CONFIG_0

MASTER_CONFIG selects the MASTER function's mode of operation, in this case:

- 0 for 500 µA usage
- 1 for smaller current usage when driving.

Offset: 0x274 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	0x0	UHSIC_PWR_P0: enables UHSIC p0 low power mode
2	0x0	UTMIP_PWR_P2: enables UTMIP p2 low power mode
1	0x0	UTMIP_PWR_P1: enables UTMIP p1 low power mode
0	0x0	UTMIP_PWR_P0: enables UTMIP p0 low power mode

9.2.157 APBDEV_PMC_TD_PWRGATE_INTER_PART_TIMER_0

TD Power-Gating Timing Between Partition Power-Gating

Offset: 0x278 │ Read/Write: R/W │ Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111)

Bit	Reset	Description
3:0	0xf	DATA: timing between consecutive partitions

9.2.158 APBDEV_PMC_UTMIP_UHSIC2_TRIGGERS_0

PMC trigger events for the UTMIP ports as well as the UHSIC port. Sleep walk clearing returns the walk pointer to 00, the first entry in a USB line walk. This is the only way to return a pointer to 0.

Pad configuration capture a set of FLLS parameters prior to resting the port so their value can be retained in deep sleep.

The trigger should only happen on the fields that are set to TRIG (1) when writing. Returned read value should always be all NULL fields. The FORCE_WALK bits trigger a SLEEP

This does not need to be a real register, so it should not consume much area. Any value read back should be 0.

Triggers for USB Ports

Offset: 0x27c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3	X	UHSIC_CLR_WAKE_ALARM_P1: Clear wake event for UHSIC port 0 0 = NULL 1 = TRIG
2	X	UHSIC_FORCE_WALK_P1: Force pointer walk for UHSIC port 0 0 = NULL 1 = TRIG
1	X	UHSIC_RESERVED_P1: Reserved for UHSIC port 0 0 = NULL 1 = TRIG
0	X	UHSIC_CLR_WALK_PTR_P1: Clear sleep walk pointer for UHSIC port 1 0 = NULL 1 = TRIG

9.2.159 APBDEV_PMC_UTMIP_UHSIC2_SAVED_STATE_0

Save some critical information about USB port prior to entering DPD in a suspend state.

SPEED

- HS: Suspend DPD was entered from a high speed mode, restore high speed at DPD exit
- FS: Suspend DPD was entered from a full speed mode, restore full speed at DPD exit
- LS: Suspend DPD was entered from a low speed mode, restore low speed at DPD exit
- RST: Reset Port, Hardware reset or DPD was not entered suspended bus, Reset and Re-enumerate port

SCRATCH -- Save other critical information about the port, software choice. Reset value should read 0x0F0F0F0F, reset should occur on Cold Hardware Reset only

Saved DPD State for all UTMIP and UHSIC Ports

Offset: 0x280 │ Read/Write: R/W │ Reset: 0x00000007 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000111)

Bit	Reset	Description
7	0x0	UHSIC_WAKE_EX_P1: Wake up on anything except a particular line value 0 = OFF 1 = ON
6	0x0	UHSIC_IGNORE_MASTER_CFG_P1
5:1	0x3	UHSIC_SCRATCH_P1
0	0x1	UHSIC_MODE_P1: UHSIC Speed prior to DPD 0 = HS 1 = RST

9.2.160 APBDEV_PMC_UTMIP_UHSIC2_SLEEP_CFG_0

This register configures the value of the line that could cause a wake-up event.

- ANY: signifies any line change DM EDGE or DP EDGE for UTMIP.
- NONE: signifies no wake-up possible.

One can use the two most significant bits (3:2) of the WAKE_VAL to 4x1 select whether:

- 00: Check both bits for value on two LSB (1:0).
- 01: Check bit DM (or strobe for HSIC) against bit 0
- 10: Check bit DP (or DATA for HSIC) against bit 1
- 11: Check for edge on DP against bit 1, edge on DM against bit 0

Sleep Walk Sequence Enables

Offset: 0x284 │ Read/Write: R/W │ Reset: 0x000000c0 (0bxxxxxxxxxxxxxxxxxxxxxxxx11000000)

Bit	Reset	Description
7:4	0xc	UHSIC_WAKE_VAL_P1: Line Value Wake Up Condition on UHSIC P1 0 = SD00 1 = SD01 2 = SD10 3 = SD11 4 = S0 5 = S1 8 = D0 10 = D1 12 = NONE

Bit	Reset	Description
		13 = SEDGE 14 = DEDGE 15 = ANY
3:1	0x0	UHSIC_RESERVED_P1: Reserved for later use for UHSIC P1
0	0x0	UHSIC_MASTER_ENABLE_P1: Enable use of master pins on UHSIC P1

9.2.161 APBDEV_PMC_UTMIP_UHSIC2_SLEEPWALK_CFG_0

This register determines when sleep walking should take effect: upon a specific GPIO event, on any wake event, or on a line value change.

Note that it is also possible to force a sleep walk: see register UTMIP_UHSIC_TRIGGERS to force the event.

Sleep Walk Sequence Enables

Offset: 0x288 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	UHSIC_LINEVAL_WALK_EN_P1: Perform walk on USB line value wake up for UHSIC P1
6	0x0	UHSIC_WAKE_WALK_EN_P1: Perform walk on any chip wake up event for UHSIC P1
5	0x0	UHSIC_GPIO_WALK_EN_P1: Perform walk on associated GPIO event for UHSIC P1
4:0	0x0	UHSIC_DESIGNATED_GPIO_P1: GPIO Number associated with UHSIC P1

9.2.162 APBDEV_PMC_UHSIC2_SLEEPWALK_P1_0

This register holds the sequence of control values to the UTMIP pads on the four consecutive cycles of a walk. The pad pins that are controlled are:

- STROBE_RPD
- DATA_RPD
- STROBE_RPU
- DATA_RPU

At this time there are no STROBE, DATA ports that should be driven by the host. So these outputs from the module should be left dangling (unconnected) until a time that feature may be implemented.

Signaling Sequence for UHSIC Port 0 Wakeup

Offset: 0x28c │ Read/Write: R/W │ Reset: 0x06060606 (0b00000110000001100000011000000110)

Bit	Reset	Description
31:28	0x0	RESERVED_D: Phase D Unused Bit
27	0x0	UHSIC_DATA_RPU_D: Phase D Pull up on DATA Line
26	0x1	UHSIC_STROBE_RPU_D: Phase D Pull Up on STROBE Line
25	0x1	UHSIC_DATA_RPD_D: Phase D Pull Down on DATA Line
24	0x0	UHSIC_STROBE_RPD_D: Phase D Pull Down on STROBE Line
23:20	0x0	RESERVED_C: Phase C Unused Bit
19	0x0	UHSIC_DATA_RPU_C: Phase C Pull up on DATA Line
18	0x1	UHSIC_STROBE_RPU_C: Phase C Pull Up on STROBE Line

Bit	Reset	Description
17	0x1	UHSIC_DATA_RPD_C: Phase C Pull Down on DATA Line
16	0x0	UHSIC_STROBE_RPD_C: Phase C Pull Down on STROBE Line
15:12	0x0	RESERVED_B: Phase B Unused Bit
11	0x0	UHSIC_DATA_RPU_B: Phase B Pull up on DATA Line
10	0x1	UHSIC_STROBE_RPU_B: Phase B Pull Up on STROBE Line
9	0x1	UHSIC_DATA_RPD_B: Phase B Pull Down on DATA Line
8	0x0	UHSIC_STROBE_RPD_B: Phase B Pull Down on STROBE Line
7:4	0x0	RESERVED_A: Phase A Unused Bit
3	0x0	UHSIC_DATA_RPU_A: Phase A Pull up on DATA Line
2	0x1	UHSIC_STROBE_RPU_A: Phase A Pull Up on STROBE Line
1	0x1	UHSIC_DATA_RPD_A: Phase A Pull Down on DATA Line
0	0x0	UHSIC_STROBE_RPD_A: Phase A Pull Down on STROBE Line

9.2.163 APBDEV_PMC_UTMIP_UHSIC2_STATUS_0

Read-only register that provides current walk pointer information as well as the line value.

Status of UTMIP UHSIC Wakeup Circuitry

Offset: 0x290 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
4	X	UHSIC_WAKE_ALARM_P1: A wake event occurred on UHSIC port 1
3	X	DATA_VAL_P1: Value of DATA line detector for UHSIC port 1
2	X	STROBE_VAL_P1: Value of STROBE line detector for UHSIC port 1
1:0	X	UHSIC_WALK_PTR_P1: Walk pointer for UHSIC port 0

9.2.164 APBDEV_PMC_UTMIP_UHSIC2_FAKE_0

Instead of using the pad value for USBOP_VAL, USBON_VAL, STROBE_VAL, DATA_VAL, VBUS_WAKEUP, ID from the UTMIP and HSIC pads, force a 2x1 mux at the input of the PMC pad macro when fake values are enabled. This could the useful feature of putting a line at rest when needed. It also has important debug merits. This mux should be placed in front of the NP synchronizer for these signals (saved power on the synchronizer). This will have to be waived in our sync checks.

Fake the Line Value for the PMC Pad Macro

Offset: 0x294 │ Read/Write: R/W │ Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0001)

Bit	Reset	Description
3	0x0	UHSIC_FAKE_DATA_EN_P1: Enable the fake line value for DATA for the PMC pad macro for UHSIC P0
2	0x0	UHSIC_FAKE_STROBE_EN_P1: Enable the fake line value for STROBE for the PMC pad macro for UHSIC P0
1	0x0	UHSIC_FAKE_DATA_VAL_P1: Fake line value for DATA for the PMC pad macro for UHSIC P0
0	0x1	UHSIC_FAKE_STROBE_VAL_P1: Fake line value for STROBE for the PMC pad macro for UHSIC P0

9.2.165 APBDEV_PMC_UTMIP_UHSIC2_LINE_WAKEUP_0

Offset: 0x298 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	UHSIC_LINE_WAKEUP_EN_P1: enables latching line wake-up event on UHSIC P1

9.2.166 APBDEV_PMC_UTMIP_MASTER2_CONFIG_0

MASTER_CONFIG selects the MASTER functions mode of operation, in this case 0 for 500uA usage, 1 for smaller current usage when driving.

Offset: 0x29c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	UHSIC_PWR_P1: enables UHSIC P1 low power mode

9.2.167 APBDEV_PMC_UTMIP_UHSIC_RPD_CFG_0

Offset: 0x2a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8	0x0	WEAKPD_ANYTIME_P2
7	0x0	DP_WEAKPD_CFG_P2
6	0x0	DM_WEAKPD_CFG_P2
5	0x0	WEAKPD_ANYTIME_P1
4	0x0	DP_WEAKPD_CFG_P1
3	0x0	DM_WEAKPD_CFG_P1
2	0x0	WEAKPD_ANYTIME_P0
1	0x0	DP_WEAKPD_CFG_P0
0	0x0	DM_WEAKPD_CFG_P0

9.2.168 APBDEV_PMC_PG_MASK_CE0_0

Offset: 0x2a4 | Read/Write: R/W | Reset: 0x000000ff (0bxxxxxxxxxxxxxxxxxxxxxxxx11111111)

Bit	Reset	Description
7:0	0xff	MASK: Mask CE1 rail

9.2.169 APBDEV_PMC_PG_MASK_3_0

Offset: 0x2a8 | Read/Write: R/W | Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:24	0xff	DISB: Mask DISB rail
23:16	0xff	DIS: Mask DIS rail
15:8	0xff	C1NC: Mask C1NC rail
7:0	0xff	C0NC: Mask C0NC rail

9.2.170 APBDEV_PMC_PG_MASK_4_0

Offset: 0x2ac | Read/Write: R/W | Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:24	0xff	SOR: Mask SOR rail
23:16	0xff	XUSBC: Mask XUSBC rail
15:8	0xff	XUSBB: Mask XUSBB rail
7:0	0xff	XUSBA: Mask XUSBA rail

9.2.171 APBDEV_PMC_PLLM_WB0_OVERRIDE2_0

Offset: 0x2b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx00000000000000000000000000)

Bit	Reset	Description
27	0x0	DIV2: New divide by 2
26	0x0	KVCO: KVCO/VCO gain
25:24	0x0	KCP: Charge pump control
23:0	0x0	SETUP: Setup

9.2.172 APBDEV_PMC_TSC_MULT_0

Offset: 0x2b4 | Read/Write: R/W | Reset: 0x000016e0 (0bxxxxxxxxxxxx000x0001011011100000)

Bit	R/W	Reset	Description
19:17	RW	0x0	TICK_SEL: This bit selects one of the six binary time stamp counter bits. 0 = BIT0 1 = BIT1 2 = BIT2 3 = BIT3 4 = BIT4 5 = BIT5
16	RO	X	FREQ_STS: Clock frequency being used for counter. 0 = Fast (i.e., oscillator frequency). 1 = Slow (i.e., 32 kHz). When the PMC_DPD_ENABLE[TSC_MULT_EN] bit is set to '1', the PMC waits for first rising edge on the slow clock before switching the counter to the slow clock. This status is also set at the first rising edge of the slow clock (to indicate the change of clock for counter).
15:0	RW	0x16e0	MULT_VAL: TSC multiply value (default based on 12 MHz oscillator).Value is (osc-freq * 16 / 32.768 kHz) when the oscillator is disabled and the TSC runs at the 32 kHz clock. For example, for a 12 MHz oscillator, VALUE = 5856.

9.2.173 APBDEV_PMC_CPU_VSENSE_OVERRIDE_0

Offset: 0x2b8 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx011111)

Bit	Reset	Description
5	0x0	VDD: VDD sensing override.
4	0x1	C0NC: C0NC VVDD partition sensing override.
3	0x1	CE3:CE3 VVDD partition sensing override.
2	0x1	CE2:CE2 VVDD partition sensing override.

Bit	Reset	Description
1	0x1	CE1:CE1 VVDD partition sensing override.
0	0x1	CE0:CE0 VVDD partition sensing override.

9.2.174 APBDEV_PMC_GLB_AMAP_CFG_0

This register configures some of the address apertures (in CCPLEX-AXD) to be MMIO or DRAM. Each bit is used to configurable one of the address aperture. For each bit 0=>MMIO and 1=>DRAM.

By default, all configurable apertures are MMIO address space. But they can be configured (at boot time) to be DRAM by programming this register. This register can be write-disabled (by the PMC_SEC_DISABLE register). The bits of this register are used as follows.

GLB_AMAP_CFG

Offset: 0x2bc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxx00000000000000000)

Bit	Reset	Description
17	0x0	IROM_HIVEC aperture: Defunct 0 = MMIO 1 = DRAM
16	0x0	AHB_A2_RSVD aperture: 0 = MMIO 1 = DRAM
15	0x0	AHB_A1_RSVD aperture: 0 = MMIO 1 = DRAM
14	0x0	AHB_A1 aperture: 0 = MMIO 1 = DRAM
13	0x0	APB_RSVD aperture: 0 = MMIO 1 = DRAM
12	0x0	EXTIO_RSVD aperture: 0 = MMIO 1 = DRAM
11	0x0	PPSB_RSVD aperture: 0 = MMIO 1 = DRAM
10	0x0	GART_GPU aperture: 0 = MMIO 1 = DRAM
9	0x0	GFX_HOST_RSVD aperture: 0 = MMIO 1 = DRAM
8	0x0	VERIF_RSVD aperture: 0 = MMIO 1 = DRAM
7	0x0	NOR_A3 aperture: 0 = MMIO 1 = DRAM
6	0x0	NOR_A2 aperture: 0 = MMIO 1 = DRAM
5	0x0	NOR_A1 aperture:

Bit	Reset	Description
		0 = MMIO 1 = DRAM
4	0x0	IRAM_RSVD aperture: 0 = MMIO 1 = DRAM
3	0x0	PCIE_A3 aperture: 0 = MMIO 1 = DRAM
2	0x0	PCIE_A2 aperture: 0 = MMIO 1 = DRAM
1	0x0	PCIE_A1 aperture: 0 = MMIO 1 = DRAM
0	0x0	IROM_LOVEC aperture: 0 = MMIO 1 = DRAM

9.2.175 APBDEV_PMC_STICKY_BITS_0

Offset: 0x2c0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8	0x0	VI: This bit is set to '1' only by a write of '1'. All other writes are ignored. Once it is set to '1', it remains '1' and only resets by a system reset (same reset that resets the main PMC).This bit is sent to the VI via a sideband signal.
7	0x0	CDD_EN: Customer Denver DFD Enable. This bit can only be written to during secure_boot_mode. It is reset by the main PMC reset. This bit value drives the cust_denver_dfd_en signal to CCPLEX. 0: DFD disabled 1: DFD enabled
6	0x0	JTAG_STS: Secure Sticky One bit to propagate JTAG enable/disable across LP0.This bit is set to '1' only by a secure write of '1'. All other writes are ignored. Once it is set to '1', it remains '1' and only resets by a system reset (same reset that resets the main PMC). 0 = ENABLE 1 = DISABLE
5	0x0	VDE4:See VDE0 description
4	0x0	VDE3:See VDE0 description
3	0x0	VDE2:See VDE0 description
2	0x0	VDE1:See VDE0 description
1	0x0	VDE0: This bit is set to '1' only by a write of '1'. All other writes are ignored. Once it is set to '1', it remains '1' and only resets by system reset (same resets which resets main PMC).This bit is send to VDE via a sideband signal.
0	0x0	HDA_LPBK_DIS: Sticky one bit to disable the loopback in HDA codec. This bit is set to ‘1’ only by a write of ‘1’. All other writes are ignored. Once it is set to ‘1’, it remains ‘1’ and only resets by system reset (the same reset which resets the main PMC).

9.2.176 APBDEV_PMC_SEC_DISABLE2_0

Offset: 0x2c4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	READ23: disable reads from secure register 23 0 = OFF 1 = ON
30	0x0	WRITE23: disable writes to secure register 23 0 = OFF 1 = ON
29	0x0	READ22: disable reads from secure register 22 0 = OFF 1 = ON
28	0x0	WRITE22: disable writes to secure register 22 0 = OFF 1 = ON
27	0x0	READ21: disable reads from secure register 21 0 = OFF 1 = ON
26	0x0	WRITE21: disable writes to secure register 21 0 = OFF 1 = ON
25	0x0	READ20: disable reads from secure register 20 0 = OFF 1 = ON
24	0x0	WRITE20: disable writes to secure register 20 0 = OFF 1 = ON
23	0x0	READ19: disable reads from secure register 19 0 = OFF 1 = ON
22	0x0	WRITE19: disable writes to secure register 19 0 = OFF 1 = ON
21	0x0	READ18: disable reads from secure register 18 0 = OFF 1 = ON
20	0x0	WRITE18: disable writes to secure register 18 0 = OFF 1 = ON
19	0x0	READ17: disable reads from secure register 17 0 = OFF 1 = ON
18	0x0	WRITE17: disable writes to secure register 17 0 = OFF 1 = ON
17	0x0	READ16: disable reads from secure register 16 0 = OFF 1 = ON
16	0x0	WRITE16: disable writes to secure register 16 0 = OFF 1 = ON
15	0x0	READ15: disable reads from secure register 15 0 = OFF 1 = ON

Bit	Reset	Description
14	0x0	WRITE15: disable writes to secure register 15 0 = OFF 1 = ON
13	0x0	READ14: disable reads from secure register 14 0 = OFF 1 = ON
12	0x0	WRITE14: disable writes to secure register 14 0 = OFF 1 = ON
11	0x0	READ13: disable reads from secure register 13 0 = OFF 1 = ON
10	0x0	WRITE13: disable writes to secure register 13 0 = OFF 1 = ON
9	0x0	READ12: disable reads from secure register 12 0 = OFF 1 = ON
8	0x0	WRITE12: disable writes to secure register 12 0 = OFF 1 = ON
7	0x0	READ11: disable reads from secure register 11 0 = OFF 1 = ON
6	0x0	WRITE11: disable writes to secure register 11 0 = OFF 1 = ON
5	0x0	READ10: disable reads from secure register 10 0 = OFF 1 = ON
4	0x0	WRITE10: disable writes to secure register 10 0 = OFF 1 = ON
3	0x0	READ9: disable reads from secure register 9 0 = OFF 1 = ON
2	0x0	WRITE9: disable writes to secure register 9 0 = OFF 1 = ON
1	0x0	READ8: disable reads from secure register 8 0 = OFF 1 = ON
0	0x0	WRITE8: disable writes to secure register 8 0 = OFF 1 = ON

9.2.177 APBDEV_PMC_WEAK_BIAS_0

Offset: 0x2c8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxx000000000000000000)

Bit	Reset	Description
17:16	0x0	XM0_DATA1: mem0_data1_vttgen_4_6_pad, mem0_data1_vttgen_5_7_pad
15:14	0x0	XM0_ADDR1: mem0_addr1_vttgen_0_pad, mem0_addr2_vttgen_0_pad
13:12	0x0	XM0_CLK_B: Defunct

Bit	Reset	Description
11:10	0x0	XM0_DATA0: mem0_data0_vttgen_0_2_pad, mem0_data0_vttgen_1_3_pad
9:8	0x0	EMMC
7:6	0x0	XM1_EXCLK: Defunct
5:4	0x0	XM1_CLK: Defunct
3:2	0x0	XM0_ADDR0: mem0_addr0_vttgen_0_pad,mem0_addr3_vttgen_0_pad
1:0	0x0	XM0_CLK: weak_bias controls for CH0 VTTGEN for clk - mem0_mclk0_vttgen_0_pad

9.2.178 APBDEV_PMC_REG_SHORT_0

Offset: 0x2cc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19	0x0	DPD_VAL_XM1_DATA: Defunct
18	0x0	DPD_VAL_XM1_ADDR1: Defunct
17	0x0	DPD_VAL_XM0_DATA1: reg_short value during DPD operation for mem0_data1_vttgen_4_6_pad, mem0_data1_vttgen_5_7_pad
16	0x0	DPD_VAL_XM0_ADDR1: reg_short value during DPD operation for mem0_addr1_vttgen_0_pad, mem0_addr2_vttgen_0_pad
15	0x0	VAL_XM1_DATA: Defunct
14	0x0	VAL_XM1_ADDR1: Defunct
13	0x0	VAL_XM0_DATA1: reg_short value during normal operation for mem0_data1_vttgen_4_6_pad, mem0_data1_vttgen_5_7_pad
12	0x0	VAL_XM0_ADDR1: reg_short value during normal operation for mem0_addr1_vttgen_0_pad, mem0_addr2_vttgen_0_pad
11	0x0	DPD_VAL_XM0_CLK_B: reg_short value during DPD operation for mem0_mclk1_vttgen_0_pad - DEFUNCT, this pad was removed
10	0x0	DPD_VAL_XM0_DATA0: reg_short value during DPD operation for mem0_data0_vttgen_0_2_pad, mem0_data0_vttgen_1_3_pad
9	0x0	VAL_XM0_CLK_B: reg_short value during normal operation for mem0_mclk1_vttgen_0_pad - DEFUNCT, this pad was removed
8	0x0	VAL_XM0_DATA0: reg_short value during normal operation for mem0_data0_vttgen_0_2_pad, mem0_data0_vttgen_1_3_pad
7	0x0	DPD_VAL_XM1_EXCLK: Defunct
6	0x0	DPD_VAL_XM1_CLK: Defunct
5	0x0	DPD_VAL_XM0_ADDR0: reg_short value during DPD operation for mem0_addr0_vttgen_0_pad,mem0_addr3_vttgen_0_pad
4	0x0	DPD_VAL_XM0_CLK: reg_short value during DPD operation for mem0_mclk0_vttgen_0_pad
3	0x0	VAL_XM1_EXCLK: Defunct
2	0x0	VAL_XM1_CLK: Defunct
1	0x0	VAL_XM0_ ADDR0: reg_short value during normal operation for mem0_addr0_vttgen_0_pad,mem0_addr3_vttgen_0_pad
0	0x0	VAL_XM0_CLK: reg_short value during normal operation for mem0_mclk0_vttgen_0_pad

9.2.179 APBDEV_PMC_PG_MASK_ANDOR_0

Power Gate Mask AND (for force power-gating) or OR (for force power-ungating) the mask value comes from the corresponding PG_MASK* register

Offset: 0x2d0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxx00000000000x0000000x00000)

Bit	Reset	Description
24	0x0	IRAM: 0 = AND 1 = OR
23	0x0	VIC: 0 = AND 1 = OR
22	0x0	XUSBC: Used as AND or OR override for XUSBC. 0 = AND 1 = OR
21	0x0	XUSBB: Used as AND or OR override for XUSBB. 0 = AND 1 = OR
20	0x0	XUSBA: Used as AND or OR override for XUSBA. 0 = AND 1 = OR
19	0x0	DISB: Used as AND or OR override for DISB partition. 0 = AND 1 = OR
18	0x0	DIS: Used as AND or OR override for DIS partition. 0 = AND 1 = OR
17	0x0	SOR: 0 = AND 1 = OR
16	0x0	C1NC: Used as AND or OR override for Cluster 1 Non-CPU partition. 0 = AND 1 = OR
15	0x0	C0NC: Used as AND or OR override for Cluster 0 Non-CPU partition. 0 = AND 1 = OR
14	0x0	CE0: Used as AND or OR override for CPU0 partition. 0 = AND 1 = OR
12	0x0	CELP: Used as AND or OR override for CELP partition. 0 = AND 1 = OR
11	0x0	CE3: Used as AND or OR override for CE3 partition. 0 = AND 1 = OR
10	0x0	CE2: Used as AND or OR override for CE2 partition. 0 = AND 1 = OR
9	0x0	CE1: Used as AND or OR override for CE1 partition. 0 = AND 1 = OR
8	0x0	SAX: 0 = AND 1 = OR

Bit	Reset	Description
7	0x0	HEG: Used as AND or OR override for HEG partition. 0 = AND 1 = OR
6	0x0	MPE: Used as AND or OR override for MPE partition. 0 = AND 1 = OR
4	0x0	VDE: Used as AND or OR override for VDE0 partition. 0 = AND 1 = OR
3	0x0	PCX: 0 = AND 1 = OR
2	0x0	VE: Used as AND or OR override for VE partition. 0 = AND 1 = OR
1	0x0	TD: Used as AND or OR override for TD. 0 = AND 1 = OR Note: The PMC generates sleep enables that are active High (that is, 1 = power on).
0	0x0	RESERVED: 0 = AND 1 = OR

9.2.180 APBDEV_PMC_GPU_RG_CNTRL_0

GPU Rail Gating Register

Offset: 0x2d4 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	RAIL_CLAMP: Enabling and removing GPU-SOC clamps 0 = DISABLE 1 = ENABLE

9.2.181 APBDEV_PMC_SEC_DISABLE3_0

Offset: 0x2d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23	0x0	READ35: disable reads from secure register 35 0 = OFF 1 = ON
22	0x0	WRITE35: disable writes to secure register 35 0 = OFF 1 = ON
21	0x0	READ34: disable reads from secure register 34 0 = OFF 1 = ON
20	0x0	WRITE34: disable writes to secure register 34 0 = OFF 1 = ON
19	0x0	READ33: disable reads from secure register 33 0 = OFF 1 = ON

Bit	Reset	Description
18	0x0	WRITE33: disable writes to secure register 33 0 = OFF 1 = ON
17	0x0	READ32: disable reads from secure register 32 0 = OFF 1 = ON
16	0x0	WRITE32: disable writes to secure register 32 0 = OFF 1 = ON
15	0x0	READ31: disable reads from secure register 31 0 = OFF 1 = ON
14	0x0	WRITE31: disable writes to secure register 31 0 = OFF 1 = ON
13	0x0	READ30: disable reads from secure register 30 0 = OFF 1 = ON
12	0x0	WRITE30: disable writes to secure register 30 0 = OFF 1 = ON
11	0x0	READ29: disable reads from secure register 29 0 = OFF 1 = ON
10	0x0	WRITE29: disable writes to secure register 29 0 = OFF 1 = ON
9	0x0	READ28: disable reads from secure register 28 0 = OFF 1 = ON
8	0x0	WRITE28: disable writes to secure register 28 0 = OFF 1 = ON
7	0x0	READ27: disable reads from secure register 27 0 = OFF 1 = ON
6	0x0	WRITE27: disable writes to secure register 27 0 = OFF 1 = ON
5	0x0	READ26: disable reads from secure register 26 0 = OFF 1 = ON
4	0x0	WRITE26: disable writes to secure register 26 0 = OFF 1 = ON
3	0x0	READ25: disable reads from secure register 25 0 = OFF 1 = ON
2	0x0	WRITE25: disable writes to secure register 25 0 = OFF 1 = ON
1	0x0	READ24: disable reads from secure register 24 0 = OFF 1 = ON

Bit	Reset	Description
0	0x0	WRITE24: disable writes to secure register 24 0 = OFF 1 = ON

9.2.182 APBDEV_PMC_SECURE_SCRATCH8_0

Secure Scratch Register

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

Bit	Reset	Description
31:0	X	SECURE_SCRATCH8

9.2.183 APBDEV_PMC_SECURE_SCRATCH9_0

Secure Scratch Register

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

Bit	Reset	Description
31:0	X	SECURE_SCRATCH9

9.2.184 APBDEV_PMC_SECURE_SCRATCH10_0

Secure Scratch Register

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

Bit	Reset	Description
31:0	X	SECURE_SCRATCH10

9.2.185 APBDEV_PMC_SECURE_SCRATCH11_0

Secure Scratch Register

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

Bit	Reset	Description
31:0	X	SECURE_SCRATCH11

9.2.186 APBDEV_PMC_SECURE_SCRATCH12_0

Secure Scratch Register

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

Bit	Reset	Description
31:0	X	SECURE_SCRATCH12

9.2.187 APBDEV_PMC_SECURE_SCRATCH13_0

Secure Scratch Register

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

Bit	Reset	Description
31:0	X	SECURE_SCRATCH13

9.2.188 APBDEV_PMC_SECURE_SCRATCH14_0

Secure Scratch Register

Offset: 0x318 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH14

9.2.189 APBDEV_PMC_SECURE_SCRATCH15_0

Secure Scratch Register

Offset: 0x31c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH15

9.2.190 APBDEV_PMC_SECURE_SCRATCH16_0

Secure Scratch Register

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

Bit	Reset	Description
31:0	X	SECURE_SCRATCH16

9.2.191 APBDEV_PMC_SECURE_SCRATCH17_0

Secure Scratch Register

Offset: 0x324 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH17

9.2.192 APBDEV_PMC_SECURE_SCRATCH18_0

Secure Scratch Register

Offset: 0x328 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH18

9.2.193 APBDEV_PMC_SECURE_SCRATCH19_0

Secure Scratch Register

Offset: 0x32c │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH19

9.2.194 APBDEV_PMC_SECURE_SCRATCH20_0

Secure Scratch Register

Offset: 0x330 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH20

9.2.195 APBDEV_PMC_SECURE_SCRATCH21_0

Secure Scratch Register

Offset: 0x334 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH21

9.2.196 APBDEV_PMC_SECURE_SCRATCH22_0

Secure Scratch Register

Offset: 0x338 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH22

9.2.197 APBDEV_PMC_SECURE_SCRATCH23_0

Secure Scratch Register

Offset: 0x33c │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH23

9.2.198 APBDEV_PMC_SECURE_SCRATCH24_0

Secure Scratch Register

Offset: 0x340 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH24

9.2.199 APBDEV_PMC_SECURE_SCRATCH25_0

Secure Scratch Register

Offset: 0x344 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH25

9.2.200 APBDEV_PMC_SECURE_SCRATCH26_0

Secure Scratch Register

Offset: 0x348 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH26

9.2.201 APBDEV_PMC_SECURE_SCRATCH27_0

Secure Scratch Register

Offset: 0x34c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH27

9.2.202 APBDEV_PMC_SECURE_SCRATCH28_0

Secure Scratch Register

Offset: 0x350 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH28

9.2.203 APBDEV_PMC_SECURE_SCRATCH29_0

Secure Scratch Register

Offset: 0x354 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH29

9.2.204 APBDEV_PMC_SECURE_SCRATCH30_0

Secure Scratch Register

Offset: 0x358 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH30

9.2.205 APBDEV_PMC_SECURE_SCRATCH31_0

Secure Scratch Register

Offset: 0x35c │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH31

9.2.206 APBDEV_PMC_SECURE_SCRATCH32_0

Secure scratch register

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

Bit	Reset	Description
31:0	X	SECURE_SCRATCH32

9.2.207 APBDEV_PMC_SECURE_SCRATCH33_0

Secure scratch register

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

Bit	Reset	Description
31:0	X	SECURE_SCRATCH33

9.2.208 APBDEV_PMC_SECURE_SCRATCH34_0

Secure scratch register

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

Bit	Reset	Description
31:0	X	SECURE_SCRATCH34

9.2.209 APBDEV_PMC_SECURE_SCRATCH35_0

Secure scratch register

Offset: 0x36c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH35

9.2.210 APBDEV_PMC_CNTRL2_0

Wake interrupt, LP0BB, and Some Miscellaneous Controls

Offset: 0x440 | Read/Write: R/W | Reset: 0x00001000 (0bxxxxxxxxxxxxxxxxxx0100xxxxxxxxxx)

Bit	Reset	Description
13	0x0	KB_SYSCLK_PM_CFG: This register bit muxes between net kb_row3 and SYS_CLK_REQ and drives that as the net kb_row3 to the I/O pad. The default setting of this bit is to select SYS_CLK_REQ. This bit is

Bit	Reset	Description
		only modified in clamshell by Boot Loader. 0 = DISABLE 1 = ENABLE
12	0x1	HOLD_CKE_LOW_EN: Enable for the PMC to assert CKE low 0 = DISABLE 1 = ENABLE
11	0x0	SYSCLK_DATA: SYS_CLK_REQ data value. This is used when SYSCLK_ORRIDE is set to '1'
10	0x0	SYSCLK_ORRIDE: SYS_CLK_REQ data override mux control 0: data is driven by hardware 1: data is driven by SYSCLK_DATA 0 = DISABLE 1 = ENABLE

9.2.211 APBDEV_PMC_IO_DPD3_REQ_0

DPD Request 3 Register

Offset: 0x45c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:30	0x0	CODE: code of operation for all set bits 0 = IDLE 1 = DPD_OFF 2 = DPD_ON
29:28	0x0	NEW_RAILS: Reserved for future use
27	0x0	DDR_DATA1_VTTGEN: mem0_data1_vttgen_4_6_pad, mem0_data1_vttgen_5_7_pad 0 = OFF 1 = ON
26	0x0	DDR_DATA0_VTTGEN: mem0_data0_vttgen_0_2_pad, mem0_data0_vttgen_1_3_pad 0 = OFF 1 = ON
25	0x0	DDR_CLKBUF1: mem0_addr2_clkbuf_1_pad, mem0_addr1_clkbuf_0_pad, mem0_addr2_clkbuf_0_pad 0 = OFF 1 = ON
24	0x0	DDR_CLKBUF0: mem0_addr0_clkbuf_1_pad, mem0_addr0_clkbuf_0_pad, mem0_addr3_clkbuf_0_pad 0 = OFF 1 = ON
23	0x0	DDR_ADDR1_VTTGEN: mem0_addr1_vttgen_0_pad,mem0_addr2_vttgen_0_pad 0 = OFF 1 = ON
22	0x0	DDR_ADDR0_VTTGEN: mem0_addr0_vttgen_0_pad,mem0_addr3_vttgen_0_pad 0 = OFF 1 = ON
21	0x0	DDR_CLK_B_VTTGEN: mem0_mclk1_vttgen_0_pad - DEFUNCT 0 = OFF 1 = ON
20	0x0	DDR_CLK_VTTGEN: mem0_mclk0_vttgen_0_pad 0 = OFF 1 = ON
19	0x0	DDR_CLK_B: xm2_mclkb_pad 0 = OFF 1 = ON

Bit	Reset	Description
18	0x0	DDR_CLK: xm2_mclk_pad 0 = OFF 1 = ON
17	0x0	DISC_BIAS: mem0_bias_0_pad 0 = OFF 1 = ON
16	0x0	DDR_MCS_B1: xm2_mcs_b_1_pad 0 = OFF 1 = ON
15	0x0	DDR_MCS_B0: xm2_mcs_b_0_pad 0 = OFF 1 = ON
14	0x0	DDR_MCKE_B1: xm2_mcke_b_1_pad 0 = OFF 1 = ON
13	0x0	DDR_MCKE_B0: xm2_mcke_b_0_pad 0 = OFF 1 = ON
12	0x0	DDR_ODT_B1: xm2_odt_b_1_pad 0 = OFF 1 = ON
11	0x0	DDR_ODT_B0: xm2_odt_b_0_pad 0 = OFF 1 = ON
10	0x0	DDR_ADDR1: xm2_ma[10,11,12,13,14,15]_pad, xm2_mba[0,1,2]_pad, xm2_ma_b[3,4,5]_pad 0 = OFF 1 = ON
9	0x0	DDR_ADDR0: xm2_ma[0,1,2,3,4,5,6,7,8,9]_pad, xm2_mras_n_pad, xm2_mcas_n_pad, xm2_mwe_n_pad 0 = OFF 1 = ON
8	0x0	DDR_RESET: xm2_reset_pad 0 = OFF 1 = ON
7	0x0	DDR_MCS1: xm2_mcs_1_pad 0 = OFF 1 = ON
6	0x0	DDR_MCS0: xm2_mcs_0_pad 0 = OFF 1 = ON
5	0x0	DDR_MCKE1: xm2_mcke_1_pad 0 = OFF 1 = ON
4	0x0	DDR_MCKE0: xm2_mcke_0_pad 0 = OFF 1 = ON
3	0x0	DDR_ODT1: xm2_odt_1_pad 0 = OFF 1 = ON
2	0x0	DDR_ODT0: xm2_odt_0_pad 0 = OFF 1 = ON
1	0x0	DDR_DATA1: xm2_data(0,1,2,3)_pad 0 = OFF

Bit	Reset	Description
		1 = ON
0	0x0	DDR_DATA0: xm2_data(4,5,6,7)_pad 0 = OFF 1 = ON

9.2.212 APBDEV_PMC_IO_DPD3_STATUS_0

DPD Status 3 Register

Offset: 0x460 | Read/Write: R/W | Reset: 0x000dfeff (0bxx00000000011011111111011111111)

Bit	Reset	Description
29:28	0x0	NEW_RAILS: Reserved for future use
27	0x0	DDR_DATA1_VTTGEN:mem0_data1_vttgen_4_6_pad, mem0_data1_vttgen_5_7_pad 0 = OFF 1 = ON
26	0x0	DDR_DATA0_VTTGEN:mem0_data0_vttgen_0_2_pad, mem0_data0_vttgen_1_3_pad 0 = OFF 1 = ON
25	0x0	DDR_CLKBUF1:mem0_addr2_clkbuf_1_pad, mem0_addr1_clkbuf_0_pad, mem0_addr2_clkbuf_0_pad 0 = OFF 1 = ON
24	0x0	DDR_CLKBUF0:mem0_addr0_clkbuf_1_pad, mem0_addr0_clkbuf_0_pad, mem0_addr3_clkbuf_0_pad 0 = OFF 1 = ON
23	0x0	DDR_ADDR1_VTTGEN:mem0_addr1_vttgen_0_pad,mem0_addr2_vttgen_0_pad 0 = OFF 1 = ON
22	0x0	DDR_ADDR0_VTTGEN:mem0_addr0_vttgen_0_pad,mem0_addr3_vttgen_0_pad 0 = OFF 1 = ON
21	0x0	DDR_CLK_B_VTTGEN:mem0_mclk1_vttgen_0_pad - DEFUNCT 0 = OFF 1 = ON
20	0x0	DDR_CLK_VTTGEN:mem0_mclk0_vttgen_0_pad 0 = OFF 1 = ON
19	0x1	DDR_CLK_B:xm2_mclkb_pad 0 = OFF 1 = ON
18	0x1	DDR_CLK:xm2_mclk_pad 0 = OFF 1 = ON
17	0x0	DISC_BIAS:mem0_bias_0_pad 0 = OFF 1 = ON
16	0x1	DDR_MCS_B1:xm2_mcs_b_1_pad 0 = OFF 1 = ON
15	0x1	DDR_MCS_B0:xm2_mcs_b_0_pad 0 = OFF 1 = ON
14	0x1	DDR_MCKE_B1:xm2_mcke_b_1_pad

Bit	Reset	Description
		0 = OFF 1 = ON
13	0x1	DDR_MCKE_B0:xm2_mcke_b_0_pad 0 = OFF 1 = ON
12	0x1	DDR_ODT_B1:xm2_odt_b_1_pad 0 = OFF 1 = ON
11	0x1	DDR_ODT_B0:xm2_odt_b_0_pad 0 = OFF 1 = ON
10	0x1	DDR_ADDR1:xm2_ma[10,11,12,13,14,15]_pad, xm2_mba[0,1,2]_pad, xm2_ma_b[3,4,5]_pad 0 = OFF 1 = ON
9	0x1	DDR_ADDR0:xm2_ma[0,1,2,3,4,5,6,7,8,9]_pad, xm2_mras_n_pad, xm2_mcas_n_pad, xm2_mwe_n_pad 0 = OFF 1 = ON
8	0x0	DDR_RESET:xm2_reset_pad 0 = OFF 1 = ON
7	0x1	DDR_MCS1:xm2_mcs_1_pad 0 = OFF 1 = ON
6	0x1	DDR_MCS0:xm2_mcs_0_pad 0 = OFF 1 = ON
5	0x1	DDR_MCKE1:xm2_mcke_1_pad 0 = OFF 1 = ON
4	0x1	DDR_MCKE0:xm2_mcke_0_pad 0 = OFF 1 = ON
3	0x1	DDR_ODT1:xm2_odt_1_pad 0 = OFF 1 = ON
2	0x1	DDR_ODT0:xm2_odt_0_pad 0 = OFF 1 = ON
1	0x1	DDR_DATA1:xm2_data(0,1,2,3)_pad 0 = OFF 1 = ON
0	0x1	DDR_DATA0:xm2_data(4,5,6,7)_pad 0 = OFF 1 = ON

9.2.213 APBDEV_PMC_STRAPPING_OPT_A_0

Strapping Options Register

Offset: 0x464 | Read/Write: R/W | Reset: 0xXXX000X0 (0bxxxxxxxxxxxxxxxxxxxxxxx00xxxxxxx0)

Bit	Reset	Description
29:26	X	BOOT_SELECT: read at power-on reset time from gmi_ad[3:0] strap pads.

Bit	Reset	Description
		0 = MPCORE_G 1 = MPCORE_LP
25	X	BOOT_SRC_USB_RECOVERY_MODE: read at power-on reset time from gmi_oe_n strap pad 0 = DISABLED 1 = ENABLED
24	X	BOOT_SRC_NOR_BOOT: read at power-on reset time from gmi_wr_n strap pad 0 = IROM 1 = NOR
23:22	X	ARM_JTAG: read at power-on reset time from gmi_ad[19:18] strap pads 00=Serial_JTAG 01=CPU_only 10=COP_only 11=Serial_JTAG(same as 00 case) 0 = SERIAL 1 = CPU 2 = COP 3 = SERIAL_ALT
9	0x0	BOOT_FAST_UART: UART Boot speed from TMC JTAG config bit 0=57600 baud 1=Osc Frequency (1 bit per OSC clock) 0 = SLOW 1 = FAST
8	RSVD1	MIO_WIDTH: 0 = RSVD1 1 = RSVD2
7:4	X	RAM_CODE: read at power-on reset time from gmi_ad[7:4] strap pads. 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 = LPDDR3_800 1 = ELPIDA_LPDDR2 2 = RSVD2 3 = DDR3_2GB 4 = RSVD4 5 = DDR3_2GB_NO_DSR 6 = DDR3_934 7 = DDR3_800 8 = RSVD8 9 = DDR3_x64_2GB 10 = DDR3_X64_8GB_667 11 = DDR3_X64_8GB_800
0	RSVD1	NOR_WIDTH: 0 = RSVD1 1 = RSVD2

9.2.214 APBDEV_PMC_SCRATCH56_0

Scratch register

Offset: 0x600 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH56: General purpose register storage

9.2.215 APBDEV_PMC_SCRATCH57_0

Scratch register

Offset: 0x604 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH57: General purpose register storage

9.2.216 APBDEV_PMC_SCRATCH58_0

Scratch register

Offset: 0x608 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH58: General purpose register storage

9.2.217 APBDEV_PMC_SCRATCH59_0

Scratch register

Offset: 0x60c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH59: General purpose register storage

9.2.218 APBDEV_PMC_SCRATCH60_0

Scratch register

Offset: 0x610 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH60: General purpose register storage

9.2.219 APBDEV_PMC_SCRATCH61_0

Scratch register

Offset: 0x614 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH61: General purpose register storage

9.2.220 APBDEV_PMC_SCRATCH62_0

Scratch register

Offset: 0x618 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH62: General purpose register storage

9.2.221 APBDEV_PMC_SCRATCH63_0

Scratch register

Offset: 0x61c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH63: General purpose register storage

9.2.222 APBDEV_PMC_SCRATCH64_0

Scratch register

Offset: 0x620 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH64: General purpose register storage

9.2.223 APBDEV_PMC_SCRATCH65_0

Scratch register

Offset: 0x624 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH65: General purpose register storage

9.2.224 APBDEV_PMC_SCRATCH66_0

Scratch register

Offset: 0x628 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH66: General purpose register storage

9.2.225 APBDEV_PMC_SCRATCH67_0

Scratch register

Offset: 0x62c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH67: General purpose register storage

9.2.226 APBDEV_PMC_SCRATCH68_0

Scratch Register

Offset: 0x630 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH68: General-purpose register storage

9.2.227 APBDEV_PMC_SCRATCH69_0

Scratch Register

Offset: 0x634 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH69: General-purpose register storage

9.2.228 APBDEV_PMC_SCRATCH70_0

Scratch Register

Offset: 0x638 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH70: General-purpose register storage

9.2.229 APBDEV_PMC_SCRATCH71_0

Scratch Register

Offset: 0x63c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH71: General-purpose register storage

9.2.230 APBDEV_PMC_SCRATCH72_0

Scratch Register

Offset: 0x640 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH72: General-purpose register storage

9.2.231 APBDEV_PMC_SCRATCH73_0

Scratch Register

Offset: 0x644 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH73: General-purpose register storage

9.2.232 APBDEV_PMC_SCRATCH74_0

Scratch Register

Offset: 0x648 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH74: General-purpose register storage

9.2.233 APBDEV_PMC_SCRATCH75_0

Scratch Register

Offset: 0x64c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH75: General-purpose register storage

9.2.234 APBDEV_PMC_SCRATCH76_0

Scratch Register

Offset: 0x650 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH76: General-purpose register storage

9.2.235 APBDEV_PMC_SCRATCH77_0

Scratch Register

Offset: 0x654 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH77: General-purpose register storage

9.2.236 APBDEV_PMC_SCRATCH78_0

Scratch Register

Offset: 0x658 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH78: General-purpose register storage

9.2.237 APBDEV_PMC_SCRATCH79_0

Scratch Register

Offset: 0x65c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH79: General-purpose register storage

9.2.238 APBDEV_PMC_SCRATCH80_0

Scratch Register

Offset: 0x660 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH80: General-purpose register storage

9.2.239 APBDEV_PMC_SCRATCH81_0

Scratch Register

Offset: 0x664 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH81: General-purpose register storage

9.2.240 APBDEV_PMC_SCRATCH82_0

Scratch Register

Offset: 0x668 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH82: General-purpose register storage

9.2.241 APBDEV_PMC_SCRATCH83_0

Scratch Register

Offset: 0x66c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH83: General-purpose register storage

9.2.242 APBDEV_PMC_SCRATCH84_0

Scratch Register

Offset: 0x670 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH84: General-purpose register storage

9.2.243 APBDEV_PMC_SCRATCH85_0

Scratch Register

Offset: 0x674 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH85: General-purpose register storage

9.2.244 APBDEV_PMC_SCRATCH86_0

Scratch Register

Offset: 0x678 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH86: General-purpose register storage

9.2.245 APBDEV_PMC_SCRATCH87_0

Scratch Register

Offset: 0x67c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH87: General-purpose register storage

9.2.246 APBDEV_PMC_SCRATCH88_0

Scratch Register

Offset: 0x680 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH88: General-purpose register storage

9.2.247 APBDEV_PMC_SCRATCH89_0

Scratch Register

Offset: 0x684 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH89: General-purpose register storage

9.2.248 APBDEV_PMC_SCRATCH90_0

Scratch Register

Offset: 0x688 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH90: General-purpose register storage

9.2.249 APBDEV_PMC_SCRATCH91_0

Scratch Register

Offset: 0x68c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH91: General-purpose register storage

9.2.250 APBDEV_PMC_SCRATCH92_0

Scratch Register

Offset: 0x690 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH92: General-purpose register storage

9.2.251 APBDEV_PMC_SCRATCH93_0

Scratch Register

Offset: 0x694 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH93: General-purpose register storage

9.2.252 APBDEV_PMC_SCRATCH94_0

Scratch Register

Offset: 0x698 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH94: General-purpose register storage

9.2.253 APBDEV_PMC_SCRATCH95_0

Scratch Register

Offset: 0x69c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH95: General-purpose register storage

9.2.254 APBDEV_PMC_SCRATCH96_0

Scratch Register

Offset: 0x6a0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH96: General-purpose register storage

9.2.255 APBDEV_PMC_SCRATCH97_0

Scratch Register

Offset: 0x6a4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH97: General-purpose register storage

9.2.256 APBDEV_PMC_SCRATCH98_0

Scratch Register

Offset: 0x6a8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH98: General-purpose register storage

9.2.257 APBDEV_PMC_SCRATCH99_0

Scratch Register

Offset: 0x6ac | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH99: General-purpose register storage

9.2.258 APBDEV_PMC_SCRATCH100_0

Scratch Register

Offset: 0x6b0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH100: General-purpose register storage

9.2.259 APBDEV_PMC_SCRATCH101_0

Scratch Register

Offset: 0x6b4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH101: General-purpose register storage

9.2.260 APBDEV_PMC_SCRATCH102_0

Scratch Register

Offset: 0x6b8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH102: General-purpose register storage

9.2.261 APBDEV_PMC_SCRATCH103_0

Scratch Register

Offset: 0x6bc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH103: General-purpose register storage

9.2.262 APBDEV_PMC_SCRATCH104_0

Scratch Register

Offset: 0x6c0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH104: General-purpose register storage

9.2.263 APBDEV_PMC_SCRATCH105_0

Scratch Register

Offset: 0x6c4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH105: General-purpose register storage

9.2.264 APBDEV_PMC_SCRATCH106_0

Scratch Register

Offset: 0x6c8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH106: General-purpose register storage

9.2.265 APBDEV_PMC_SCRATCH107_0

Scratch Register

Offset: 0x6cc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH107: General-purpose register storage

9.2.266 APBDEV_PMC_SCRATCH108_0

Scratch Register

Offset: 0x6d0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH108: General-purpose register storage

9.2.267 APBDEV_PMC_SCRATCH109_0

Scratch Register

Offset: 0x6d4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH109: General-purpose register storage

9.2.268 APBDEV_PMC_SCRATCH110_0

Scratch Register

Offset: 0x6d8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH110: General-purpose register storage

9.2.269 APBDEV_PMC_SCRATCH111_0

Scratch Register

Offset: 0x6dc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH111: General-purpose register storage

9.2.270 APBDEV_PMC_SCRATCH112_0

Scratch Register

Offset: 0x6e0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH112: General-purpose register storage

9.2.271 APBDEV_PMC_SCRATCH113_0

Scratch Register

Offset: 0x6e4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH113: General-purpose register storage

9.2.272 APBDEV_PMC_SCRATCH114_0

Scratch Register

Offset: 0x6e8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH114: General-purpose register storage

9.2.273 APBDEV_PMC_SCRATCH115_0

Scratch Register

Offset: 0x6ec | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH115: General-purpose register storage

9.2.274 APBDEV_PMC_SCRATCH116_0

Scratch Register

Offset: 0x6f0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH116: General-purpose register storage

9.2.275 APBDEV_PMC_SCRATCH117_0

Scratch Register

Offset: 0x6f4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH117: General-purpose register storage

9.2.276 APBDEV_PMC_SCRATCH118_0

Scratch Register

Offset: 0x6f8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH118: General-purpose register storage

9.2.277 APBDEV_PMC_SCRATCH119_0

Scratch Register

Offset: 0x6fc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH119: General-purpose register storage

9.2.278 APBDEV_PMC_SCRATCH1_ECO_0

Scratch register

Offset: 0x700 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	Reserved

9.3 PMC Counter 0 Registers

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

9.3.1 SYSCTR0_CNTCR_0

The control registers are read/written by secure accesses only except the CNTFID0 and COUNTERID11-0 registers, which can be written only once by secure or non-secure access.

The CNTFID0 and COUNTERID11-0 registers are read-only; however, to initialize them at boot (by the ARM7 or main CPU), they are defined as write once.

A non-secure write to the control registers is ignored. A non-secure read to the control registers returns all 1s.

Counter Control Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxx00)

Bit	Reset	Description
8	0x0	FCREQ: Requested frequency modes table entry. This bit is not used.
1	0x0	HDBG: Halt-on-debug. Controls whether a Halt-on-debug signal halts the system counter or not. 0: System counter ignores Halt-on-debug.1: Asserted Halt-on-debug signal halts the system counter update. 0 = DISABLE 1 = ENABLE
0	0x0	EN: Enables the counter. When this bit is written to '1', then the TSC is loaded with the CNTCVx register’s value, and starts incrementing from that value. When this bit is written to '0', the TSC halts counting and stays at that value. 0: System counter disabled 1: System counter enabled. 0 = DISABLE 1 = ENABLE

9.3.2 SYSCTR0_CNTSR_0

Counter Status Register

Offset: 0x4 | Read/Write: RO | Reset: 0x00000X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8	X	FCREQ: Frequency change acknowledge. This bit is a copy of the value of the CNTCR[FCREQ] bit.
1	X	HDBG: Indicates whether or not the counter is halted because the Halt-on-Debug signal is asserted: 0: Counter is not halted. 1: Counter is halted. 0 = DISABLE 1 = ENABLE
0	X	RESERVED: Reserved, no impact.

9.3.3 SYSCTR0_CNTCV0_0

Counter Count Value [31:0] Register

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

Bit	Reset	Description
31:0	0x0	CV: Counter value [31:0]. A read of this register provides the TSC[31:0] value at the time of the read. When CNTCR[EN]=0, a write to this register is used to initialize the TSC[31:0] value. When

Bit	Reset	Description
		CNTCR[EN]=1, a write to this register has an unpredictable behavior.

9.3.4 SYSCTR0_CNTCV1_0

Counter Count Value[63:32] Register

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

Bit	Reset	Description
31:0	0x0	CV: Counter value [63:32]. A read of this register provides the TSC[63:32] value at the time of the read. When CNTCR[EN]=0, a write to this register is used to initialize the TSC[63:32] value. When CNTCR[EN]=1, a write to this register has an unpredictable behavior.

9.3.5 SYSCTR0_CNTFID0_0

Frequency Table Entry Register

Offset: 0x20 | Read/Write: R/W | Reset: 0x00b71b00 (0b00000000101101110001101100000000)

Bit	Reset	Description
31:0	0xb71b00	FV: Counter frequency value in Hz. The default is set to 12 MHz. This register can be written only once.

9.3.6 SYSCTR0_CNTFID1_0

Frequency Table End Marker

Offset: 0x24 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	FV: Counter frequency value end marker (all 0s, read-only).

9.3.7 SYSCTR0_COUNTERID4_0

The COUNTERID11-0 registers are read-only; however, to initialize them at boot (by the ARM7™ or main CPU), they are defined as write once.

Offset: 0xfd0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VALUE: Peripheral ID value

9.3.8 SYSCTR0_COUNTERID5_0

Offset: 0xfd4 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	RESERVED: Reserved. A read will return all 0s.

9.3.9 SYSCTR0_COUNTERID6_0

Offset: 0xfd8 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	RESERVED: Reserved. A read will return all 0s.

9.3.10 SYSCTR0_COUNTERID7_0

Offset: 0xfdc | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	RESERVED: Reserved. A read will return all 0s.

9.3.11 SYSCTR0_COUNTERID0_0

Offset: 0xfe0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VALUE: Peripheral ID value

9.3.12 SYSCTR0_COUNTERID1_0

Offset: 0xfe4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VALUE: Peripheral ID value

9.3.13 SYSCTR0_COUNTERID2_0

Offset: 0xfe8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VALUE: Peripheral ID value

9.3.14 SYSCTR0_COUNTERID3_0

Offset: 0xfec | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VALUE: Peripheral ID value

9.3.15 SYSCTR0_COUNTERID8_0

Offset: 0xff0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VALUE: Component ID value

9.3.16 SYSCTR0_COUNTERID9_0

Offset: 0xff4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
7:0	0x0	VALUE: Component ID value

9.3.17 SYSCTR0_COUNTERID10_0

Offset: 0xff8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VALUE: Component ID value

9.3.18 SYSCTR0_COUNTERID11_0

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

Bit	Reset	Description
7:0	0x0	VALUE: Component ID value

9.4 PMC Counter 1 Registers

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

The status registers are some control registers readable via the CNTReadBase base address by secure or non-secure accesses.

These are the same physical registers that are described in the Control Register section.

9.4.1 SYSCTR1_CNTCV0_0

Counter Count Value [31:0] Register

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

Bit	Reset	Description
31:0	X	CV: Counter value [31:0]

9.4.2 SYSCTR1_CNTCV1_0

Counter Count Value [63:32] Register

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

Bit	Reset	Description
31:0	X	CV: Counter value [63:32]

9.4.3 SYSCTR1_COUNTERID4_0

Offset: 0xfd0 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: Peripheral ID value.

9.4.4 SYSCTR1_COUNTERID5_0

Offset: 0xfd4 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	RESERVED: Reserved. A read will return all 0s.

9.4.5 SYSCTR1_COUNTERID6_0

Offset: 0xfd8 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	RESERVED: Reserved. A read will return all 0s.

9.4.6 SYSCTR1_COUNTERID7_0

Offset: 0xfdc | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	RESERVED: Reserved. A read will return all 0s.

9.4.7 SYSCTR1_COUNTERID0_0

Offset: 0xfe0 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: Peripheral ID value

9.4.8 SYSCTR1_COUNTERID1_0

Offset: 0xfe4 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: Peripheral ID value

9.4.9 SYSCTR1_COUNTERID2_0

Offset: 0xfe8 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: Peripheral ID value

9.4.10 SYSCTR1_COUNTERID3_0

Offset: 0xfec | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: Peripheral ID value

9.4.11 SYSCTR1_COUNTERID8_0

Offset: 0xff0 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: Component ID value

9.4.12 SYSCTR1_COUNTERID9_0

Offset: 0xff4 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: Component ID value

9.4.13 SYSCTR1_COUNTERID10_0

Offset: 0xff8 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: Component ID value

9.4.14 SYSCTR1_COUNTERID11_0

Offset: 0xffc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VALUE: Component ID value

9.5 Secure Boot Registers

9.5.1 SB_CSR_0

Secure Boot Control Status Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x0000002X (0bxxxxxxxxxxxxxxxx000000000010xx01)

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

Bit	R/W	Reset	Description
			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

9.5.2 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: 0x4 | Read/Write: R/W | Reset: 0x00001000 (0b00000000000000000001000000000000)

Bit	Reset	Description
31:0	0x1000	PROTECTED_ROM_START: PROTECTED_ROM_START

9.5.3 SB_PFCFG_0

Secure Boot Processor Feature Configuration Register

Offset: 0x8 | Read/Write: R/W | Reset: 0x00XX008f (0bxxxxxxxx0110xxxxxxxx0000010001111)

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

Bit	R/W	Reset	Description
			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
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 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

9.5.4 SB_SECURE_SPAREREG_0_0

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

Bit	Reset	Description
31:0	0x0	SECURE_SPAREREG_0

9.5.5 SB_SECURE_SPAREREG_1_0

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SECURE_SPAREREG_1

9.5.6 SB_SECURE_SPAREREG_2_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SECURE_SPAREREG_2

9.5.7 SB_SECURE_SPAREREG_3_0

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

Bit	Reset	Description
31:0	0x0	SECURE_SPAREREG_3

9.5.8 SB_SECURE_SPAREREG_4_0

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

Bit	Reset	Description
31:0	0x0	SECURE_SPAREREG_4

9.5.9 SB_SECURE_SPAREREG_5_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SECURE_SPAREREG_5

9.5.10 SB_SECURE_SPAREREG_6_0

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

Bit	Reset	Description
31:0	0x0	SECURE_SPAREREG_6

9.5.11 SB_SECURE_SPAREREG_7_0

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

Bit	Reset	Description

Bit	Reset	Description
31:0	0x0	SECURE_SPAREREG_7

[THIS PAGE INTENTIONALLY LEFT BLANK]

10.0 ACTIVITY MONITOR

This section describes the activity monitor (ACTMON) block. The ACTMON acts as a central repository for activity indicators from various units, performs averaging of past samples, and indicates any abnormality in activity levels by sending an interrupt to software. Devices provide an idle/busy signal to the ACTMON block.

10.1 Functional Description

Activity monitoring is a means to dynamically measure the utilization of units in the system to help drive software power management policies. In Tegra processors, Dynamic Voltage and Frequency Scaling (DVFS) is used as a power management mechanism. It uses utilization information from various units to select an efficient voltage and frequency that the unit should operate at, while providing the requisite performance. Voltage change decisions are handled by this central software agent, but the unit frequency can be modulated in one of the following ways:

- The unit controls its own frequency - no need for activity monitoring support.
- The device driver controls the unit frequency based on calculated workload or hints from a user-level driver – no need for activity monitoring support.
- The device driver controls the unit frequency based on utilization information from a “unit ACTMON”.
- The central DVFS agent controls the unit frequency based on utilization information from a “central ACTMON”.

In addition to tracking utilization, activity monitoring is also used to provide hardware support to make the process of activity monitoring interrupt-driven instead of polling-based. This support is in the form of averaging, watermark detection, and histogram hardware. This reduces software overhead in procuring and tracking utilization data.

The ACTMON block has the following features:

- One 32-bit counter for each signaling device that accumulates activity counts.
- Running average computation for each counter over past N samples (N=128).
- Watermark breach detection logic for each counter. This is used to detect abrupt change in utilization.
- Interrupt-driven operation.

The DVFS thread can directly access and use the activity monitors to compute the target frequency (and consequently, voltage) for each unit for the next sampling interval.

The following table summarizes the Tegra K1 activity monitors.

Table 30: Activity Monitors

Module	Counter Description (Counter Width = 32 Bits)
A7AVP	Number of system clock cycles that A7AVP is in the halt state
AHB	Number of AHB clock cycles when no data transfer is detected on the bus. Can select one specific master or all of them.
APB	Number of APB clock cycles when no data transfer is detected on the bus. Can select one specific master or all of them.
CPU	The flow controller asserts a signal when ALL the cores are halted.
MC-ALL	Number of Memory Controller clock cycles when any (including CCPLEX) memory access event is detected. The MC toggles a signal to ACTMON every ‘W’ cycles.
MC-CPU	Number of MC clock cycles when any memory access event from CCPLEX is detected. MC toggles a signal to ACTMON every ‘W’ cycles.

Module	Counter Description (Counter Width = 32 Bits)
VDE	Number of video pipe cycles when the pipe is empty.

The table above assumes that the following units either run at a fixed frequency or do their own frequency management based on fixed workload, so they do not need central ACTMON support.

- VI/CSI (fixed)
- Display (workload-based, no monitors needed)
- Host (fixed)
- ISP (workload-based)
- MSENC/VIC (utilization-based and uses per-unit ACTMON)
- SEC (fixed frequency)
- USB3 –Falcon (workload-based)
- MSELECT (frequency is based on other units and cannot be controlled independently)

10.2 ACTMON Registers

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

10.2.1 ACTMON GLB Status Register

10.2.1.1 ACTMON_GLB_STATUS_0

Offset: 0x0 | Read/Write: RO | Reset: 0xXX00XX00 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CPU_INTR: CPU Monitor Interrupt status. 1 = Interrupt detected; 0 = Interrupt not detected 0 = NOINTR 1 = INTR
30	X	COP_INTR: COP Monitor Interrupt status. 1 = Interrupt detected; 0 = Interrupt not detected 0 = NOINTR 1 = INTR
29	X	AHB_INTR: AHB Monitor Interrupt status. 1 = Interrupt detected; 0 = Interrupt not detected 0 = NOINTR 1 = INTR
28	X	APB_INTR: APB Monitor Interrupt status. 1 = Interrupt detected; 0 = Interrupt not detected 0 = NOINTR 1 = INTR
27	X	VDE_INTR: VDE Interrupt status. 1 = Interrupt detected; 0 = Interrupt not detected 0 = NOINTR 1 = INTR
26	X	MCALL_INTR: MC_ALL Interrupt status. 1 = Interrupt detected; 0 = Interrupt not detected 0 = NOINTR 1 = INTR
25	X	MCCPU_INTR: MC_CPU Interrupt status. 1 = Interrupt detected; 0 = Interrupt not detected 0 = NOINTR : 1 = INTR
15	X	CPU_MON_ACT: CPU monitor active status. 1 = active 0 = INACTIVE 1 = ACTIVE

Bit	Reset	Description
14	X	COP_MON_ACT: COP monitor active status. 1 = active 0 = INACTIVE 1 = ACTIVE
13	X	AHB_MON_ACT: 0 = inactive. 1 = active 0 = INACTIVE 1 = ACTIVE
12	X	APB_MON_ACT: 0 = inactive. 1 = active 0 = INACTIVE 1 = ACTIVE
10	X	VDE_MON_ACT: 0 = inactive. 1 = active 0 = INACTIVE 1 = ACTIVE
9	X	MCALL_MON_ACT: 0 = inactive. 1 = active 0 = INACTIVE 1 = ACTIVE
8	X	MCCPU_MON_ACT: 0 = inactive. 1 = active 0 = INACTIVE 1 = ACTIVE

10.2.2 ACTMON Global Period Register

10.2.2.1 ACTMON_GLB_PERIOD_CTRL_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8	0x0	SOURCE: 0 = MSEC : Sampling period time base in milliseconds 1 = USEC : Sampling period time base in microseconds
7:0	0x0	SAMPLE_PERIOD: Sampling period in milliseconds/microseconds (implemented as n+1 counter). Programming a 0 implies a 1 millisecond/microsecond sampling period.

10.2.3 CPU Registers

10.2.3.1 ACTMON_CPU_CTRL_0

Monitor Control Register

Offset: 0x80 | Read/Write: R/W | Reset: 0x00001800 (0b00000000000000xxxxx110xxxxxxxxxx)

Bit	Reset	Description
31	0x0	ENB: Enable Monitor. Set by software to enable sampling. Cleared in one of the following ways: (a) When software intends to stop the monitor, it can do so by clearing this field, (b) when the sampling period expires (and we are not in the periodic mode) 0 = DISABLE 1 = ENABLE
30	0x0	CONSECUTIVE_ABOVE_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_UPPER_NUM' upper watermark breaches are detected 0 = DISABLE : interrupt is disabled. 1 = ENABLE: interrupt is enabled
29	0x0	CONSECUTIVE_BELOW_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_LOWER_NUM' lower watermark breaches are detected. 0 = DISABLE : interrupt is disabled. 1 = ENABLE: interrupt is enabled
28:26	0x0	CONSECUTIVE_ABOVE_WMARK_NUM: Number (N+1) of consecutive upper watermark breaches

Bit	Reset	Description
		that need to occur to raise an interrupt 0 = DISABLE : interrupt is disabled. 1 = ENABLE: interrupt is enabled
25:23	0x0	CONSECUTIVE_BELOW_WMARK_NUM: Number (N+1) of consecutive lower watermark breaches that need to occur to raise an interrupt 0 = DISABLE : interrupt is disabled. 1 = ENABLE: interrupt is enabled
22	0x0	WHEN_OVERFLOW_EN: Enable interrupt for number of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE : interrupt is disabled. 1 = ENABLE: interrupt is enabled
21	0x0	AVG_ABOVE_WMARK_EN: Enable interrupt when AVG value is above its upper watermark value. 0 = DISABLE : interrupt is disabled. 1 = ENABLE: interrupt is enabled
20	0x0	AVG_BELOW_WMARK_EN: Enable interrupt when AVG value is below its lower watermark value. 0 = DISABLE : interrupt is disabled. 1 = ENABLE: interrupt is enabled
19	0x0	AT_END_EN:Enable interrupt at the end of sample period. 0 = DISABLE : interrupt is disabled. 1 = ENABLE: interrupt is enabled
18	0x0	ENB_PERIODIC: Enable periodic mode. Sample for one sample period or periodic sampling 0 = DISABLE : periodic mode is disabled, samples for only 1 period 1 = ENABLE: periodic mode is enabled, keeps on sampling and updating value after every sample period
12:10	0x6	K_VAL: variable for IIR filter. Default is 6, which translates to 2^(K+1) = 128.

10.2.3.2 ACTMON_CPU_UPPER_WMARK_0

Upper Watermark Register

Offset: 0x84 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.3.3 ACTMON_CPU_LOWER_WMARK_0

Lower Watermark Register

Offset: 0x88 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.3.4 ACTMON_CPU_INIT_AVG_0

Initial AVG value, specified by SW to set up the filter

Offset: 0x8c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL

10.2.3.5 ACTMON_CPU_AVG_UPPER_WMARK_0

AVG Upper Watermark Register

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

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.3.6 ACTMON_CPU_AVG_LOWER_WMARK_0

AVG Lower Watermark Register

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

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.3.7 ACTMON_CPU_COUNT_WEIGHT_0

Count Weight Register

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

Bit	Reset	Description
31:0	X	VAL: weight value for count measurements.

10.2.3.8 ACTMON_CPU_COUNT_0

Monitor Status0 Register

Offset: 0x9c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the number of active count pulses in one period

10.2.3.9 ACTMON_CPU_AVG_COUNT_0

Monitor Status1 Register

Offset: 0xa0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the AVG count value

10.2.3.10 ACTMON_CPU_INTR_STATUS_0

Monitor Interrupt Register

Offset: 0xa4 | Read/Write: R/W | Reset: 0x00000000 (0b000xx000xxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	CONSECUTIVE_UPPER: Assert at the end of sampling period, if count value crosses upper watermark value consecutively for the number of times specified in CONSECUTIVE_UPPER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect

Bit	Reset	Description
		0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
30	0x0	CONSECUTIVE_LOWER: Assert at the end of sampling period, if count value crosses lower watermark value consecutively for the number of times specified in CONSECUTIVE_LOWER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
29	0x0	AT_END: Assert at the end of sampling period, if interrupt at end of sample period is enabled. A write value of 1 clears this interrupt; writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
26	0x0	WHEN_OVERFLOW: Assert at the end of sampling period, if there is an overflow. A write value of 1 clears this interrupt; writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
25	0x0	AVG_BELOW_WMARK: Assert at the end of sampling period, if AVG count value crosses lower AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect. 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
24	0x0	AVG_ABOVE_WMARK: Assert at the end of sampling period, if AVG count value crosses upper AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected

10.2.4 COP Registers

10.2.4.1 ACTMON_COP_CTRL_0

Monitor Control Register

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00001800 (0b00000000000000xxxxx110xxxxxxxxxx)

Bit	Reset	Description
31	0x0	ENB: Enable Monitor. Set by software to enable sampling. Cleared in one of the following ways: (a) When software intends to stop the monitor, it can do so by clearing this field, (b) when the sampling period expires (and we are not in the periodic mode) 0 = DISABLE 1 = ENABLE
30	0x0	CONSECUTIVE_ABOVE_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_UPPER_NUM' upper watermark breaches are detected. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
29	0x0	CONSECUTIVE_BELOW_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_LOWER_NUM' lower watermark breaches are detected. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
28:26	0x0	CONSECUTIVE_ABOVE_WMARK_NUM: Number (N+1) of consecutive upper watermark breaches that need to occur to raise an interrupt. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
25:23	0x0	CONSECUTIVE_BELOW_WMARK_NUM: Number (N+1) of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
22	0x0	WHEN_OVERFLOW_EN: Enable interrupt for the Number of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE : interrupt is disabled.

Bit	Reset	Description
		1 = ENABLE : interrupt is enabled.
21	0x0	AVG_ABOVE_WMARK_EN: Enable interrupt when AVG value is above its upper watermark value. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
20	0x0	AVG_BELOW_WMARK_EN: Enable interrupt when AVG value is below its lower watermark value. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
19	0x0	AT_END_EN: Enable interrupt at the end of sample period. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
18	0x0	ENB_PERIODIC: Enable periodic mode. Sample for one sample period or periodic sampling 0 = DISABLE : periodic mode is disabled, samples for only 1 period 1 = ENABLE : periodic mode is enabled, keeps on sampling and updating value after every sample period
12:10	0x6	K_VAL: variable for IIR filter. Default is 6, which translates to 2^(K+1) = 128.

10.2.4.2 ACTMON_COP_UPPER_WMARK_0

Upper Watermark Register

Offset: 0xc4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.4.3 ACTMON_COP_LOWER_WMARK_0

Lower Watermark Register

Offset: 0xc8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.4.4 ACTMON_COP_INIT_AVG_0

Initial AVG value, specified by software to set up the filter

Offset: 0xcc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL

10.2.4.5 ACTMON_COP_AVG_UPPER_WMARK_0

AVG Upper Watermark Register

Offset: 0xd0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.4.6 ACTMON_COP_AVG_LOWER_WMARK_0

AVG Lower Watermark Register

Offset: 0xd4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.4.7 ACTMON_COP_COUNT_WEIGHT_0

Count Weight Register

Offset: 0xd8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: weight value for count measurements.

10.2.4.8 ACTMON_COP_COUNT_0

Monitor Status0 Register

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

Bit	Reset	Description
31:0	X	VAL: Indicates the number of active count pulses in one period

10.2.4.9 ACTMON_COP_AVG_COUNT_0

Monitor Status1 Register

Offset: 0xe0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the AVG count value

10.2.4.10 ACTMON_COP_INTR_STATUS_0

Monitor Interrupt Register

Offset: 0xe4 | Read/Write: R/W | Reset: 0x00000000 (0b000xx000xxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	CONSECUTIVE_UPPER: Assert at the end of sampling period, if count value crosses upper watermark value consecutively for the number of times specified in CONSECUTIVE_UPPER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
30	0x0	CONSECUTIVE_LOWER: Assert at the end of sampling period, if count value crosses lower watermark value consecutively for the number of times specified in CONSECUTIVE_LOWER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
29	0x0	AT_END: Assert at the end of sampling period, if interrupt at end of sample period is enabled. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected

Bit	Reset	Description
		1 = INTR : 1 = interrupt detected
26	0x0	WHEN_OVERFLOW: Assert at the end of sampling period, if there is an overflow. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
25	0x0	AVG_BELOW_WMARK: Assert at the end of sampling period, if AVG count value crosses lower AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
24	0x0	AVG_ABOVE_WMARK: Assert at the end of sampling period, if AVG count value crosses upper AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected

10.2.5 AHB Registers

10.2.5.1 ACTMON_AHB_CTRL_0

Monitor Control Register

Offset: 0x100 | Read/Write: R/W | Reset: 0x00001800 (0b00000000000000xxxxx1100000000000)

Bit	Reset	Description
31	0x0	ENB: Enable Monitor. Set by SW to enable sampling. Cleared in one of the following ways: (a) When software intends to stop the monitor, it can do so by clearing this field, (b) when the sampling period expires (and we are not in the periodic mode) 0 = DISABLE 1 = ENABLE
30	0x0	CONSECUTIVE_ABOVE_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_UPPER_NUM' upper watermark breaches are detected. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
29	0x0	CONSECUTIVE_BELOW_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_LOWER_NUM' lower watermark breaches are detected. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
28:26	0x0	CONSECUTIVE_ABOVE_WMARK_NUM: Number (N+1) of consecutive upper watermark breaches that need to occur to raise an interrupt. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
25:23	0x0	CONSECUTIVE_BELOW_WMARK_NUM: Number (N+1) of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
22	0x0	WHEN_OVERFLOW_EN: Enable interrupt for the Number of consecutive lower watermark breaches that need to occur to raise an interruptd. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
21	0x0	AVG_ABOVE_WMARK_EN:Enable interrupt when AVG value is above its upper watermark value. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
20	0x0	AVG_BELOW_WMARK_EN:Enable interrupt when AVG value is below its lower watermark value. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
19	0x0	AT_END_EN:Enable interrupt at the end of sample period. 0 = DISABLE : interrupt is disabled.

Bit	Reset	Description
		1 = ENABLE : interrupt is enabled.
18	0x0	ENB_PERIODIC: Enable periodic mode. Sample for one sample period or periodic sampling 0 = DISABLE : periodic mode is disabled, samples for only 1 period 1 = ENABLE : periodic mode is enabled, keeps on sampling and updating value after every sample period
12:10	0x6	K_VAL: variable for IIR filter. Default is 6, which translates to 2^(K+1) = 128.
9:4	0x0	MONITOR_COND: Selection criteria on parent module for the activity signal, used only for APB/AHB monitors. Indicates which AHB master to monitor. All 1s means any master. 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = ARC 5 = AHBDMA 6 = USB 7 = APBDMA 8 = NA1 9 = NA2 10 = NA3 11 = SNOR 12 = NA4 13 = BSEV 14 = SE 15 = DDS 16 = BSEA 17 = USB3 18 = USB2 19 = NA5 20 = NA6 21 = MIPIHSI 22 = NA7 23 = NA8 24 = NA9 63 = ALL
3:0	0x0	SAMPLE_COND: Selection criteria on parent module for type of pulse signal, used only for APB/AHB monitor. 0 = AHB_MASTER_ACTIVE 1 = AHB_MASTER_SLAVE_ACTIVE 2 = AHB_DATA_XFER 3 = AHB_IDLE 4 = MASTER_IDLE 5 = AHB_BUSY 6 = DISABLE

10.2.5.2 ACTMON_AHB_UPPER_WMARK_0

Upper Watermark Register

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

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.5.3 ACTMON_AHB_LOWER_WMARK_0

Lower Watermark Register

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

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.5.4 ACTMON_AHB_INIT_AVG_0

Initial AVG value, specified by software to set up the filter

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

Bit	Reset	Description
31:0	X	VAL

10.2.5.5 ACTMON_AHB_AVG_UPPER_WMARK_0

AVG Upper Watermark Register

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

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.5.6 ACTMON_AHB_AVG_LOWER_WMARK_0

AVG Lower Watermark Register

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

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.5.7 ACTMON_AHB_COUNT_WEIGHT_0

Count Weight Register

Offset: 0x118 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: weight value for count measurements.

10.2.5.8 ACTMON_AHB_COUNT_0

Monitor Status0 Register

Offset: 0x11c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the number of active count pulses in one period

10.2.5.9 ACTMON_AHB_AVG_COUNT_0

Monitor Status1 Register

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

Bit	Reset	Description
31:0	X	VAL: Indicates the AVG count value

10.2.5.10 ACTMON_AHB_INTR_STATUS_0

Monitor Interrupt Register

Offset: 0x124 | Read/Write: R/W | Reset: 0x00000000 (0b000xx000xxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	CONSECUTIVE_UPPER: Assert at the end of sampling period, if count value crosses upper watermark value consecutively for the number of times specified in CONSECUTIVE_UPPER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
30	0x0	CONSECUTIVE_LOWER: Assert at the end of sampling period, if count value crosses lower watermark value consecutively for the number of times specified in CONSECUTIVE_LOWER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
29	0x0	AT_END: Assert at the end of sampling period, if interrupt at end of sample period is enabled. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
26	0x0	WHEN_OVERFLOW: Assert at the end of sampling period, if there is an overflow. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
25	0x0	AVG_BELOW_WMARK: Assert at the end of sampling period, if AVG count value crosses lower AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
24	0x0	AVG_ABOVE_WMARK: Assert at the end of sampling period, if AVG count value crosses upper AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected

10.2.6 APB Registers

10.2.6.1 ACTMON_APB_CTRL_0

Monitor Control Register

Offset: 0x140 | Read/Write: R/W | Reset: 0x00001800 (0b00000000000000xxxxx1100000000000)

Bit	Reset	Description
31	0x0	ENB: Enable Monitor. Set by SW to enable sampling. Cleared in one of the following ways: (a) When SW intends to stop the monitor, it can do so by clearing this field, (b) when the sampling period expires (and we are not in the periodic mode) 0 = DISABLE 1 = ENABLE
30	0x0	CONSECUTIVE_ABOVE_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_UPPER_NUM' upper watermark breaches are detected. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
29	0x0	CONSECUTIVE_BELOW_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_LOWER_NUM' lower watermark breaches are detected. 0 = DISABLE : interrupt is disabled.

Bit	Reset	Description
		1 = ENABLE : interrupt is enabled.
28:26	0x0	CONSECUTIVE_ABOVE_WMARK_NUM: Number (N+1) of consecutive upper watermark breaches that need to occur to raise an interrupt. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
25:23	0x0	CONSECUTIVE_BELOW_WMARK_NUM: Number (N+1) of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
22	0x0	WHEN_OVERFLOW_EN: Enable interrupt for the Number of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
21	0x0	AVG_ABOVE_WMARK_EN: Enable interrupt when AVG value is above its upper watermark value. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
20	0x0	AVG_BELOW_WMARK_EN: Enable interrupt when AVG value is below its lower watermark value. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
19	0x0	AT_END_EN: Enable interrupt at the end of sample period. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
18	0x0	ENB_PERIODIC: Enable periodic mode. Sample for one sample period or periodic sampling 0 = DISABLE : periodic mode is disabled, samples for only 1 period 1 = ENABLE : periodic mode is enabled, keeps on sampling and updating value after every sample period
12:10	0x6	K_VAL: variable for IIR filter. Default is 6, which translates to 2^(K+1) = 128.
9:4	0x0	MONITOR_COND: Selection criteria on parent module for the activity signal, used only for APB/AHB monitors. Indicates which APB slave to monitor. All 1s means any slave. 0 = AUDIO 1 = UARTA 2 = UARTB 3 = UARTC 4 = UARTD 5 = VFIR 6 = DTV 7 = I2C1 8 = I2C2 9 = I2C3 10 = I2C4 11 = DVC 12 = I2C6 13 = SPI1 14 = SPI2 15 = SPI3 16 = SPI4 17 = SPI5 18 = SPI6 19 = OWR 63 = ALL
3:0	0x0	SAMPLE_COND: Selection criteria on parent module for type of pulse signal, used only for APB/AHB monitor. 0 = PSEL_ACTIVE 1 = PREADY_ACTIVE 2 = PENABLE_PSEL_ACTIVE 3 = APB_IDLE 4 = DISABLE

10.2.6.2 ACTMON_APB_UPPER_WMARK_0

Upper Watermark Register

Offset: 0x144 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.6.3 ACTMON_APB_LOWER_WMARK_0

Lower Watermark Register

Offset: 0x148 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.6.4 ACTMON_APB_INIT_AVG_0

Initial AVG value, specified by SW to set up the filter

Offset: 0x14c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL

10.2.6.5 ACTMON_APB_AVG_UPPER_WMARK_0

AVG Upper Watermark Register

Offset: 0x150 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.6.6 ACTMON_APB_AVG_LOWER_WMARK_0

AVG Lower Watermark Register

Offset: 0x154 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.6.7 ACTMON_APB_COUNT_WEIGHT_0

Count Weight Register

Offset: 0x158 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: weight value for count measurements.

10.2.6.8 ACTMON_APB_COUNT_0

Monitor Status0 Register

Offset: 0x15c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the number of active count pulses in one period

10.2.6.9 ACTMON_APB_AVG_COUNT_0

Monitor Status1 Register

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

Bit	Reset	Description
31:0	X	VAL: Indicates the AVG count value

10.2.6.10 ACTMON_APB_INTR_STATUS_0

Monitor Interrupt Register

Offset: 0x164 | Read/Write: R/W | Reset: 0x00000000 (0b000xx000xxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	CONSECUTIVE_UPPER: Assert at the end of sampling period, if count value crosses upper watermark value consecutively for the number of times specified in CONSECUTIVE_UPPER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
30	0x0	CONSECUTIVE_LOWER: Assert at the end of sampling period, if count value crosses lower watermark value consecutively for the number of times specified in CONSECUTIVE_LOWER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
29	0x0	AT_END: Assert at the end of sampling period, if interrupt at end of sample period is enabled. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
26	0x0	WHEN_OVERFLOW: Assert at the end of sampling period, if there is an overflow. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
25	0x0	AVG_BELOW_WMARK: Assert at the end of sampling period, if AVG count value crosses lower AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
24	0x0	AVG_ABOVE_WMARK: Assert at the end of sampling period, if AVG count value crosses upper AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected

10.2.6.11 ACTMON_APB_CTRL_SAPB_0

Monitor Interrupt Register

Offset: 0x168 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:4	0x0	MONITOR_COND_SAPB: Selection criteria on parent module for the activity signal, used only for APB/AHB monitors. 0 = DVFS 1 = PWM 2 = MISC 3 = RTC 4 = KBC 5 = PMC 6 = SYSCTR0 7 = SYSCTR1 8 = FUSE 9 = KFUSE 10 = LA 11 = DDS 12 = DP2 13 = SNOR 14 = SE 15 = CSITE 16 = MC 17 = EMC 18 = MIPI_CAL 19 = ATOMICS 20 = CED 21 = HDA 22 = SATA 23 = SDMMC1 24 = SDMMC2 25 = SDMMC3 26 = SDMMC4 27 = SATA_AUX 28 = APB2JTAG 29 = SOC_THERM 30 = XUSB_HOST 31 = XUSB_DEV 32 = XUSB_PADCTL 33 = PINMUX_AUX 34 = SECURE_REGS 63 = ALL
3:0	0x0	SAMPLE_COND_SAPB: Selection criteria on parent module for type of pulse signal, used only for APB/AHB monitor. 0 = PSEL_ACTIVE 1 = PREADY_ACTIVE 2 = PENABLE_PSEL_ACTIVE 3 = APB_IDLE 4 = DISABLE

10.2.7 VDE Registers

10.2.7.1 ACTMON_VDE_CTRL_0

Monitor Control Register

Offset: 0x180 | Read/Write: R/W | Reset: 0x00001800 (0b00000000000000xxxxx110xxxxxxxxxx)

Bit	Reset	Description
31	0x0	ENB: Enable Monitor. Set by software to enable sampling. Cleared in one of the following ways: (a) When SW intends to stop the monitor, it can do so by clearing this field, (b) when the sampling period expires (and we are not in the periodic mode) 0 = DISABLE 1 = ENABLE
30	0x0	CONSECUTIVE_ABOVE_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_UPPER_NUM' upper watermark breaches are detected. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.

Bit	Reset	Description
29	0x0	CONSECUTIVE_BELOW_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_LOWER_NUM' lower watermark breaches are detected. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
28:26	0x0	CONSECUTIVE_ABOVE_WMARK_NUM: Number (N+1) of consecutive upper watermark breaches that need to occur to raise an interrupt. 0 = DISABLE : interrupt is disabled. 1 = ENABLE
25:23	0x0	CONSECUTIVE_BELOW_WMARK_NUM: Number (N+1) of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
22	0x0	WHEN_OVERFLOW_EN: Enable interrupt for the Number of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
21	0x0	AVG_ABOVE_WMARK_EN: Enable interrupt when AVG value is above its upper watermark value. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
20	0x0	AVG_BELOW_WMARK_EN: Enable interrupt when AVG value is below its lower watermark value. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
19	0x0	AT_END_EN: Enable interrupt at the end of sample period. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
18	0x0	ENB_PERIODIC: Enable periodic mode. Sample for one sample period or periodic sampling 0 = DISABLE : periodic mode is disabled, samples for only 1 period 1 = ENABLE : periodic mode is enabled, keeps on sampling and updating value after every sample period
12:10	0x6	K_VAL: variable for IIR filter. Default is 6, which translates to 2^(K+1) = 128.

10.2.7.2 ACTMON_VDE_UPPER_WMARK_0

Upper Watermark Register

Offset: 0x184 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.7.3 ACTMON_VDE_LOWER_WMARK_0

Lower Watermark Register

Offset: 0x188 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.7.4 ACTMON_VDE_INIT_AVG_0

Initial AVG value, specified by SW to set up the filter

Offset: 0x18c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL

10.2.7.5 ACTMON_VDE_AVG_UPPER_WMARK_0

AVG Upper Watermark Register

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

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.7.6 ACTMON_VDE_AVG_LOWER_WMARK_0

AVG Lower Watermark Register

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

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.7.7 ACTMON_VDE_COUNT_WEIGHT_0

Count Weight Register

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

Bit	Reset	Description
31:0	X	VAL: weight value for count measurements.

10.2.7.8 ACTMON_VDE_COUNT_0

Monitor Status0 Register

Offset: 0x19c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the number of active count pulses in one period

10.2.7.9 ACTMON_VDE_AVG_COUNT_0

Monitor Status1 Register

Offset: 0x1a0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the AVG count value

10.2.7.10 ACTMON_VDE_INTR_STATUS_0

Monitor Interrupt Register

Offset: 0x1a4 | Read/Write: R/W | Reset: 0x00000000 (0b000xx000xxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	CONSECUTIVE_UPPER: Assert at the end of sampling period, if count value crosses upper watermark value consecutively for the number of times specified in CONSECUTIVE_UPPER_NUM field. a write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
30	0x0	CONSECUTIVE_LOWER: Assert at the end of sampling period, if count value crosses lower watermark value consecutively for the number of times specified in CONSECUTIVE_LOWER_NUM field. a write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
29	0x0	AT_END: Assert at the end of sampling period, if interrupt at end of sample period is enabled. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
26	0x0	WHEN_OVERFLOW: Assert at the end of sampling period, if there is an overflow. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
25	0x0	AVG_BELOW_WMARK: Assert at the end of sampling period, if AVG count value crosses lower AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
24	0x0	AVG_ABOVE_WMARK: Assert at the end of sampling period, if AVG count value crosses upper AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected

10.2.8 MC_ALL Registers

10.2.8.1 ACTMON_MCALL_CTRL_0

Monitor Control Register

Offset: 0x1c0 | Read/Write: R/W | Reset: 0x00001800 (0b00000000000000xxxxx110xxxxxxxxxx)

Bit	Reset	Description
31	0x0	ENB: Enable Monitor. Set by software to enable sampling. Cleared in one of the following ways: (a) When software intends to stop the monitor, it can do so by clearing this field, (b) when the sampling period expires (and we are not in the periodic mode) 0 = DISABLE 1 = ENABLE
30	0x0	CONSECUTIVE_ABOVE_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_UPPER_NUM' upper watermark breaches are detected. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
29	0x0	CONSECUTIVE_BELOW_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_LOWER_NUM' lower watermark breaches are detectedd. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
28:26	0x0	CONSECUTIVE_ABOVE_WMARK_NUM: Number (N+1) of consecutive upper watermark breaches that need to occur to raise an interrupt. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
25:23	0x0	CONSECUTIVE_BELOW_WMARK_NUM: Number (N+1) of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.

Bit	Reset	Description
22	0x0	WHEN_OVERFLOW_EN: Enable interrupt for the Number of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
21	0x0	AVG_ABOVE_WMARK_EN:Enable interrupt when AVG value is above its upper watermark value. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
20	0x0	AVG_BELOW_WMARK_EN:Enable interrupt when AVG value is below its lower watermark value. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
19	0x0	AT_END_EN:Enable interrupt at the end of sample period. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
18	0x0	ENB_PERIODIC: Enable periodic mode. Sample for one sample period or periodic sampling 0 = DISABLE : periodic mode is disabled, samples for only 1 period 1 = ENABLE : periodic mode is enabled, keeps on sampling and updating value after every sample period
12:10	0x6	K_VAL: variable for IIR filter. Default is 6, which translates to 2^(K+1) = 128.

10.2.8.2 ACTMON_MCALL_UPPER_WMARK_0

Upper Watermark Register

Offset: 0x1c4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.8.3 ACTMON_MCALL_LOWER_WMARK_0

Lower Watermark Register

Offset: 0x1c8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.8.4 ACTMON_MCALL_INIT_AVG_0

Initial AVG value, specified by SW to set up the filter

Offset: 0x1cc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL

10.2.8.5 ACTMON_MCALL_AVG_UPPER_WMARK_0

AVG Upper Watermark Register

Offset: 0x1d0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.8.6 ACTMON_MCALL_AVG_LOWER_WMARK_0

AVG Lower Watermark Register

Offset: 0x1d4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.8.7 ACTMON_MCALL_COUNT_WEIGHT_0

Count Weight Register

Offset: 0x1d8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: weight value for count measurements.

10.2.8.8 ACTMON_MCALL_COUNT_0

Monitor Status0 Register

Offset: 0x1dc | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the number of active count pulses in one period

10.2.8.9 ACTMON_MCALL_AVG_COUNT_0

Monitor Status1 Register

Offset: 0x1e0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the AVG count value

10.2.8.10 ACTMON_MCALL_INTR_STATUS_0

Monitor Interrupt Register

Offset: 0x1e4 | Read/Write: R/W | Reset: 0x00000000 (0b000xx000xxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	CONSECUTIVE_UPPER: Assert at the end of sampling period, if count value crosses upper watermark value consecutively for the number of times specified in CONSECUTIVE_UPPER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
30	0x0	CONSECUTIVE_LOWER: Assert at the end of sampling period, if count value crosses lower watermark value consecutively for the number of times specified in CONSECUTIVE_LOWER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
29	0x0	AT_END: Assert at the end of sampling period, if interrupt at end of sample period is enabled. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected

Bit	Reset	Description
		1 = INTR : 1 = interrupt detected
26	0x0	WHEN_OVERFLOW: Assert at the end of sampling period, if there is an overflow. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
25	0x0	AVG_BELOW_WMARK: Assert at the end of sampling period, if AVG count value crosses lower AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
24	0x0	AVG_ABOVE_WMARK: Assert at the end of sampling period, if AVG count value crosses upper AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected

10.2.9 MC_CPU Registers

10.2.9.1 ACTMON_MCCPU_CTRL_0

Monitor Control Register

Offset: 0x200 | Read/Write: R/W | Reset: 0x00001800 (0b00000000000000xxxxx110xxxxxxxxxx)

Bit	Reset	Description
31	0x0	ENB: Enable Monitor. Set by software to enable sampling. Cleared in one of the following ways: (a) When software intends to stop the monitor, it can do so by clearing this field, (b) when the sampling period expires (and we are not in the periodic mode) 0 = DISABLE 1 = ENABLE
30	0x0	CONSECUTIVE_ABOVE_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_UPPER_NUM' upper watermark breaches are detected. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
29	0x0	CONSECUTIVE_BELOW_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_LOWER_NUM' lower watermark breaches are detected. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
28:26	0x0	CONSECUTIVE_ABOVE_WMARK_NUM: Number (N+1) of consecutive upper watermark breaches that need to occur to raise an interrupt. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
25:23	0x0	CONSECUTIVE_BELOW_WMARK_NUM: Number (N+1) of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
22	0x0	WHEN_OVERFLOW_EN: Enable interrupt for the Number of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
21	0x0	AVG_ABOVE_WMARK_EN: Enable interrupt when AVG value is above its upper watermark value. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
20	0x0	AVG_BELOW_WMARK_EN: Enable interrupt when AVG value is below its lower watermark value. 0 = DISABLE : interrupt is disabled. 1 = ENABLE : interrupt is enabled.
19	0x0	AT_END_EN: Enable interrupt at the end of sample period. 0 = DISABLE : interrupt is disabled.

Bit	Reset	Description
		1 = ENABLE : interrupt is enabled.
18	0x0	ENB_PERIODIC: Enable periodic mode. Sample for one sample period or periodic sampling 0 = DISABLE : periodic mode is disabled, samples for only 1 period 1 = ENABLE : periodic mode is enabled, keeps on sampling and updating value after every sample period
12:10	0x6	K_VAL: variable for IIR filter. Default is 6, which translates to 2^(K+1) = 128.

10.2.9.2 ACTMON_MCCPU_UPPER_WMARK_0

Upper Watermark Register

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

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.9.3 ACTMON_MCCPU_LOWER_WMARK_0

Lower Watermark Register

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

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.9.4 ACTMON_MCCPU_INIT_AVG_0

Initial AVG value, specified by software to set up the filter

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

Bit	Reset	Description
31:0	X	VAL

10.2.9.5 ACTMON_MCCPU_AVG_UPPER_WMARK_0

AVG Upper Watermark Register

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

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.9.6 ACTMON_MCCPU_AVG_LOWER_WMARK_0

AVG Lower Watermark Register

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

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.9.7 ACTMON_MCCPU_COUNT_WEIGHT_0

Count Weight Register

Offset: 0x218 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: weight value for count measurements.

10.2.9.8 ACTMON_MCCPU_COUNT_0

Monitor Status0 Register

Offset: 0x21c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the number of active count pulses in one period

10.2.9.9 ACTMON_MCCPU_AVG_COUNT_0

Monitor Status1 Register

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

Bit	Reset	Description
31:0	X	VAL: Indicates the AVG count value

10.2.9.10 ACTMON_MCCPU_INTR_STATUS_0

Monitor interrupt Register

Offset: 0x224 | Read/Write: R/W | Reset: 0x00000000 (0b000xx000xxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	CONSECUTIVE_UPPER: Assert at the end of sampling period, if count value crosses upper watermark value consecutively for the number of times specified in CONSECUTIVE_UPPER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
30	0x0	CONSECUTIVE_LOWER: Assert at the end of sampling period, if count value crosses lower watermark value consecutively for the number of times specified in CONSECUTIVE_LOWER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
29	0x0	AT_END: Assert at the end of sampling period, if interrupt at end of sample period is enabled. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
26	0x0	WHEN_OVERFLOW: Assert at the end of sampling period, if there is an overflow. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected
25	0x0	AVG_BELOW_WMARK: Assert at the end of sampling period, if AVG count value crosses lower AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected

Bit	Reset	Description
24	0x0	AVG_ABOVE_WMARK: Assert at the end of sampling period, if AVG count value crosses upper AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR : 0 = interrupt not detected 1 = INTR : 1 = interrupt detected

10.2.10 Histogram Registers

10.2.10.1 ACTMON_HISTOGRAM_CONFIG_0

Histogram Configuration

Offset: 0x300 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxx0000xxxxxxxxxxx)

Bit	Reset	Description
15:12	NONE	SOURCE: 0 = NONE 1 = AHB 2 = APB 3 = COP 4 = CPU 5 = MC_ALL 6 = MC_CPU 7 = VDE 8 = VDEA 9 = APB_MMIO
8:4	X	SHIFT: Scaling factor for the idle counter before the value is used to update histogram bucket.
3	X	STALL_ON_SINGLE_SATURATE: FALSE : continue incrementing buckets even when another bucket has saturated TRUE : stop incrementing bucket when at least one other bucket has saturated 0 = FALSE 1 = TRUE
2	X	NO_UNDERFLOW_BUCKET: FALSE : increase bucket 0 when idle time is less than the minimum value TRUE : ignore idle times that are less than the minimum value 0 = FALSE 1 = TRUE
1	X	LINEAR_MODE: DISABLE : bucket width increases exponentially with a power of two ENABLE : bucket width is the same for all buckets 0 = DISABLE 1 = ENABLE
0	X	ACTIVE: ENABLE : enable histogram recording 0 = DISABLE 1 = ENABLE

10.2.10.2 ACTMON_HISTOGRAM_CTRL_0

Offset: 0x304 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	CLEAR: Clear all

10.2.10.3 ACTMON_HISTOGRAM_DATA_0

This is an array of 32 identical register entries; the register fields below apply to each entry.

Offset: 0x380..0x3ff | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VALUE

11.0 REAL-TIME CLOCK

The Real-Time Clock (RTC) module maintains seconds and milliseconds counters, and five alarm registers. The RTC is in the ‘always-on’ power domain, allowing for the counters to run and alarms to trigger when the system is in low-power state. If configured, interrupts triggered by the RTC can cause the system to wake up from a low-power state.

Features

- 10-bit milliseconds counter that runs off of a 32.768 KHz clock source.
- 32-bit seconds counter that increment for every 1000 milliseconds.
- Alarm feature that triggers an interrupt when the specified value matches the milliseconds counter.
- Alarm feature that triggers an interrupt when the specified value matches the seconds counter.
- Count-down alarm feature that triggers an alarm after counting down the specified number of seconds.
- Count-down alarm feature that triggers an alarm after counting down the specified number of milliseconds.
- Security bit that disables further processor writes to the seconds counter and ensures that the RTC clock keeps running.
- Hardware adjusts drift in clock which can occur due to PPM variations in oscillator output.

11.1 Functional Description

The RTC operates in two clock domains: the APB clock domain and the 32 KHz clock domain. The RTC continues updating the 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 the 32 KHz clock domain. Writes are transferred to the 32 KHz domain with BUSY.STATUS set to BUSY until the transfer is completed. Reads are shadowed in the APB clock domain and return immediately.

The RTC implements a Seconds Counter register, a Milliseconds Counter register, three Alarm registers, two countdown alarms, and various interrupt-related registers.

Writes to the seconds counter can be disabled by writing to the CONTROL.DIS_WR_SEC_CNT bit. Alarm registers and countdown alarm registers set interrupt bits when corresponding events occur. Interrupt status, mask, set and source registers reflect the status and also allow setting and clearing of various bits. All registers (except the BUSY register) are implemented in the 32K clock domain. Writes are transferred to the 32K domain with BUSY STATUS set to BUSY until the transfer completes. Reads are shadowed in the APB domain and will return immediately.

Resets

The RTC receives an asynchronous reset which is synchronized to the APB clock and RTC clock domains.

11.2 RTC Registers

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

11.2.1 APBDEV_RTC_CONTROL_0

Control Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	WR_SEC_CNT: When set, writes to the SECONDS counter are disabled. Can only be cleared by resetting the RTC module 0 = DISABLE 1 = ENABLE

11.2.2 APBDEV_RTC_BUSY_0

Busy Register

Offset: 0x4 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	STATUS: This bit is set when a write is initiated on the APB side. It is cleared once the write completes in RTC 32 KHz clock domain, which could be several thousands of APB clocks. This must be IDLE before a write is initiated. Note that this bit is only for writes. 0 = IDLE 1 = BUSY

11.2.3 APBDEV_RTC_SECONDS_0

Seconds Counter Register

The SECONDS register is copied over to the APB side every eight 32 KHz clocks (~250 µs). Because of this, performing a read immediately after a write might return an old value. This covers 49710.26 Days (or) 136.192 Years of 365 days each.

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

Bit	Reset	Description
31:0	0x0	SECONDS: The seconds counter is incremented every 1000 milliseconds.

11.2.4 APBDEV_RTC_SHADOW_SECONDS_0

Shadowed Seconds Counter Register

Shadow SECONDS register is updated over to the APB side whenever there is a read to milliseconds counter.

Since the software cannot read both registers at any given point of time, the Seconds register content is captured in this register. If software needs to read both seconds and milliseconds, read the MILLI_SECONDS register followed by a read of this register.

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

Bit	Reset	Description
31:0	X	SHADOW_SECONDS: A snapshot of the SECONDS counter is taken, whenever there is a read to MILLI_SECONDS Register.

11.2.5 APBDEV_RTC_MILLI_SECONDS_0

Milliseconds Counter Register

The Milliseconds register is copied over to the APB side every eight 32 KHz clocks (~250 µs). Because of this, performing a read immediately after a write might return an old value.

Offset: 0x10 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	MILLI_SECONDS: Milliseconds counter is incremented using the Bresenham algorithm

11.2.6 APBDEV_RTC_SECONDS_ALARM0_0

Seconds Alarm0 Registers

When the value in this register matches the seconds counter, the corresponding interrupt status bit is set. If enabled, the interrupt line is asserted. A value of zero disables the alarm.

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SECS_MATCH_VALUE: Match value to trigger the alarm

11.2.7 APBDEV_RTC_SECONDS_ALARM1_0

Seconds Alarm1 Registers

When the value in this register matches the seconds counter, the corresponding interrupt status bit is set. If enabled, the interrupt line is asserted. A value of zero disables the alarm.

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

Bit	Reset	Description
31:0	0x0	SECS_MATCH_VALUE: Match value to trigger the alarm

11.2.8 APBDEV_RTC_MILLI_SECONDS_ALARM_0

Milliseconds Alarm Register

When the value in this register matches the milliseconds counter, the corresponding interrupt status bit is set. If enabled, the interrupt line is asserted. A value of zero disables the alarm.

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:0	0x0	MSEC_MATCH_VALUE: Milliseconds match value.

11.2.9 APBDEV_RTC_SECONDS_COUNTDOWN_ALARM_0

Countdown Alarm Register

If ENABLE is set to ENABLED, an internal counter is loaded with VALUE and counted down. The interrupt bit corresponding to the countdown_alarm_0 is set after the specified number of seconds have elapsed.

The interrupt bit corresponding to the countdown alarm (SEC_CDN_ALARM) is set after the specified interval has expired. If REPEAT is ENABLED, the countdown operation is performed repeatedly.

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	ENABLE: Enable bit for the countdown operation. If repeat is not set, this bit is cleared once the internal counters counts down to specified value. 0 = DISABLED 1 = ENABLED
30	0x0	REPEAT: Repeat bit for the countdown operation 0 = DISABLED 1 = ENABLED
29:0	0x0	VALUE: Number of seconds to countdown

11.2.10 APBDEV_RTC_MILLI_SECONDS_COUNTDOWN_ALARM_0

Countdown Alarm Register

If ENABLE is set to ENABLED, an internal counter is loaded with VALUE and counted down. The interrupt bit corresponding to the countdown_alarm_0 is set after the specified number of milliseconds have elapsed.

The interrupt bit corresponding to the countdown alarm (MSEC_CDN_ALARM) is set after the specified interval has expired. If REPEAT is ENABLED, the countdown operation is performed repeatedly.

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

Bit	Reset	Description
31	0x0	ENABLE: Enable bit for the countdown operation. If repeat is not set, this bit is cleared once the internal counters counts down to specified value. 0 = DISABLED 1 = ENABLED
30	0x0	REPEAT: Repeat bit for the countdown operation 0 = DISABLED 1 = ENABLED
29:0	0x0	VALUE: Number of milliseconds to countdown

11.2.11 APBDEV_RTC_INTR_MASK_0

Interrupt Mask Register

This register stores the masks for the interrupts. If a bit is set 1 and the corresponding interrupt condition is satisfied, interrupt line to system interrupt controller is asserted.

Offset: 0x28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	MSEC_CDN_ALARM
3	0x0	SEC_CDN_ALARM
2	0x0	MSEC_ALARM
1	0x0	SEC_ALARM1
0	0x0	SEC_ALARM0

11.2.12 APBDEV_RTC_INTR_STATUS_0

Interrupt Status Register

Bits in this register are high after the interrupt condition is satisfied. Any bits that are set to one in the data written to this register will clear their corresponding data bits. Any bits set to zero have no effect.

Offset: 0x2c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	MSEC_CDN_ALARM
3	0x0	SEC_CDN_ALARM
2	0x0	MSEC_ALARM
1	0x0	SEC_ALARM1
0	0x0	SEC_ALARM0

11.2.13 APBDEV_RTC_INTR_SOURCE_0

Interrupt Source Register

This read-only register returns the AND of the Interrupt Source and Interrupt Mask registers.

Offset: 0x30 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
4	X	MSEC_CDN_ALARM
3	X	SEC_CDN_ALARM
2	X	MSEC_ALARM
1	X	SEC_ALARM1
0	X	SEC_ALARM0

11.2.14 APBDEV_RTC_INTR_SET_0

Interrupt Set Register

Reserved. This write-only register is used for testing purposes only.

Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	MSEC_CDN_ALARM
3	0x0	SEC_CDN_ALARM
2	0x0	MSEC_ALARM
1	0x0	SEC_ALARM1
0	0x0	SEC_ALARM0

11.2.15 APBDEV_RTC_CORRECTION_FACTOR_0

Correction Factor (Digital Trimming) Register

Support is for +/- 500 ppm. The expected maximum deviation of the standard clock sources is +/- 50 ppm.

Offset: 0x38 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	0x0	DIRECTION: 0 = DECREMENT 1 = INCREMENT
8:0	0x0	PPM

12.0 HOST SUBSYSTEM

The host controller (referred to as Host1x) provides a sophisticated programming and control interface to various graphic and video engines, and display. In addition to Host1x's control interfaces with these units, it also has a slave (IP) interface to the ARM7 crossbar (xbar), TSEC, and a direct memory interface to fetch command structures from system memory. Commands are either gathered from a push buffer in memory or provided directly by the CPU, and then supplied to the clients behind Host1x via Host1x channels. The channels also provide a means of synchronization between software and any individual block or amongst the blocks themselves via hardware Sync Point signals (syncpts).

12.1 Glossary

Term	Definition
CDMA	The command DMA. It is responsible for reading commands from memory and supplying them to all channels. It starts from the address in the channel's DMASTART register and continues until it reaches the address in the channel's DMAPUT register.
Channel	A piece of hardware that provides a programming interface to one or more classes. A channel contains a sequence of commands embodied in a command FIFO. This sequence of commands can also be thought of as a thread of execution and of a single context. There exists only one sequence of commands or context per channel. That is to say, there is no hardware-managed context switching within a single channel.
Channel Commands	Entries in the command FIFO. Commands take a common form interpretable by Host and are used for 3 main functions: controlling class ownership and class virtualization of the channel; command expansion via gather commands; and issuing class methods.
Channel switch	A transfer of ownership of a client from one channel to another; a type of context switch. Sometimes a source of confusion, this is not a transfer of ownership of a channel, but of a client. This is the preferred narrowed nomenclature to context switch in order to avoid confusion.
Class	An abstraction of a client, device, or resource. It is a collection of methods. A class can only be owned by multiple channels, but only one channel may actively be using any class, which is known as a channel's working class. The transfer of working class from one channel to another is known as a context switch, which is managed by Host.
Class ID	A unique tag corresponding to each class.
Class method	A method that belongs to an unspecified class.
Class switch	A change in the current in the active class of a client; a type of context switch. In the event that a class switch also involves a channel switch, channel switch is the preferred declaration.
Client, device, resource	A piece of hardware that resides behind Host, connected via the HRD and HRW buses. Client is the preferred terminology for this document. Generally mapped one-to-one with a class, although this is not a strict requirement.
Command FIFO	Holds channel commands supplied by the CDMA or PIO accesses. It is stalled by synchronization methods, backpressure from its destination client, or ownership of the

Term	Definition
	destination client by a different channel
Context switch	A Host event where it facilitates a client's state transition. A context switch is either a channel switch or a class switch.
Direct register access	Access to a client initiated by the CPU. If a channel and the CPU initiate a data transaction to the same client at the same time, the CPU gets priority. Direct register accesses are orthogonal to channels in regards to client ownership.
DMAGET register	Holds the current address of the CDMA. One exists per channel.
DMAPUT register	Holds the end address of a command stream in memory. One exists per channel.
DMASTART register	Holds the start address of a command stream in memory. One exists per channel.
Gather	A channel command to gather contiguous chunks of memory and inserts it into the command stream.
Host Master	General term to indicate an entity that can control Host. The current list is the CPU, AVP, and TSEC.
Host method	A method that belongs to the Host class.
Increment	A channel command that specifies an offset and a count. Increment works like nonincrement, except the offset is incremented by one on each on each write.
Indirect register access	Access to a client initiated by the CPU, but requires two Host register accesses. This is deprecated in the SOC version of the host; it comes from the companion-chip mode where fixed-latency interfaces to the CPU may exist. Like direct register accesses, indirect takes priority over channel transactions. Indirect register accesses must be tied a channel because they require a read return FIFO.
Mask	A channel command that specifies an offset and a mask. A mask command works much like increment and non-increment, except the offsets are calculated by looking at the bits set in the mask. The bit position indicates the relative offset from the specified offset in the command. Mask expects the number of words to follow to be equal to the number of bits set in the mask.
Method	An operation on a class. The most common example is a single register write.
Nonincrement	A channel command that specifies an offset and a count. The offset specifies a method in the current class and the count indicates the number of subsequent words to be written at that offset.
Protected Channel	Deprecated feature previously used by software for flow control.
Push-buffer	A set of commands residing contiguously in memory. It is a communication method between the processor and Host. Commands are placed at the end of the buffer and a pointer is updated in Host.
SetClass	A channel command that takes a class id as an argument. SetClass is the sole means to indicate the current virtualization of the channel. It also the sole means that a channel can acquire ownership of a class. Subsequent

Term	Definition
	channel commands are directed towards this class.
Sync Point (Syncpt)	A counter used for synchronization. Every time a specified event occurs, the counter is incremented. A channel can wait until the syncpt attains a specified value, or an interrupt can be generated when a specified value is reached.
Teardown	Can be either a module or channel teardown. Essentially, it means all links and references within Host between the specified module or the specified channel are removed and reset.
Tick Count	Running count of the Host clock after it has been enabled.
WAIT	A host method that takes a single vector argument. It is used in conjunction with syncpt. A WAIT stalls the command FIFO until the supplied vector intersects the RAISE register at all.

12.2 Features

The key features of the Host1x controller are summarized below:

- 12 Channels
- 192 Sync Points
- 64 Syncpt Base Registers
- 32-bit Syncpt Comparison
- 32-bit Timeout register
- Stall and transfer counters
- Unit Activity Monitor (ACTMON): MSENC and VIC
- Master clients: Crossbar and TSEC
- TZ secure bit In MMIO path
- Master Interface protocol: Native Crossbar
- Client Interface protocol: Hwr/Hrd
- 32-bit Host1x2MC address

12.2.1 Class Based Programming

GPUs support a programming interface that is based on writing to offsets within a “class” that implements a function, rather than to specific register offsets and formats.

By using “class” interface register offsets/formats need not to remain fixed so there is nothing chip specific in the API. This will allow both hardware flexibility and software driver compatibility.

12.2.2 Command Buffer DMA

To maximize the Host1x bandwidth, it is important to burst as many write cycles as possible, which requires the processor to first combine writes. The write buffering typically requires sequential offsets to be written and also may not follow strict programming order. The CPU will first store this buffered data directly to memory and then program the Host1x engine to DMA it.

12.2.3 Multiple Channels

A channel is a thread of execution within Host1x. Similar to a multithreaded CPU, a channel helps defining a context that can be used to allow multiple users of the Host1x and plays a role in context switching.

A channel switch does not always require a client module context switch. There will be a state change within the Host1x, but if channels have non-overlapping usage (e.g., possibly a VIC channel and MSENC encode channel), there may be no need to context switch any Host1x client module.

12.3 Hardware Features

12.3.1 Channels

The Tegra K1 processor has 12 Host1x channels. Each channel has a set of registers and a command FIFO. Each channel can be associated with one or more Host1x clients. The channel is the primary means of delivering commands to clients, which is described in the Host1x Programming Model section.

12.3.1.1 Channel Registers

Channel registers are broken into two functional groups: one grouping is associated with the command FIFO and command delivery to clients; the other grouping is associated with register and memory access.

CDMA Registers

- **HOST1X_CHANNEL_DMASTART_0:** This register triggers a DMA fetch from memory for this channel, if PUT register does not equal the GET register.
- **HOST1X_CHANNEL_DMAPUT_0:** This register triggers a DMA fetch from memory for this channel, if the PUT register does not equal the GET register. This address is relative to the DMASTART base address. It does not support byte writes. All 4-byte data need to be programmed.
- **HOST1X_CHANNEL_DMAGET_0:** This register tracks the MC offset, which DMA engine has read. It gets incremented as entries are loaded from the channels command buffer into the FIFO. This address is relative to the DMASTART base address.
- **HOST1X_CHANNEL_DMAEND_0:** This is the boundary of illegal addresses (either end of push-buffer or end of physical memory). This is designed to prevent DMA from prefetching illegal addresses. If DMA reaches this address before seeing a RESTART, it will stop. This would be a software error condition.
- **HOST1X_CHANNEL_DMACTRL_0:** The various fields of DMA control register are described as below:
  - **DMAGETRST**: Reset GET pointer to '0'. Useful for cleaning up crashed channels. DMAGET value is not updated instantly. It takes 4 cycles between programming of reset and valid DMAGET.
  - **DMAGETINIT**: Reset GET pointer to the value of DMAPUT when DMAGETRST is asserted.
  - **DMASTOP**: Stop DMA from fetching on this channel.

> **Note:** A Command DMA channel needs to be enabled for PIO-gather to work.

Access Registers

- **HOST1X_CHANNEL_INDOFF_0 and HOST1X_CHANNEL_INDOFF2_0:** The INDOFF and INDOFF2 registers (along with INDCNT and INDDATA) are used to indirectly read/write modules outside the host. If AUTOINC is set, INDOFFSET value is increments by 4 on every access of INDDATA. REGFNUMEMPTY is polled to determine when valid data can be read from INDDATA. The INDOFF register has limited capability on chips with large memory maps. If the top bit of the memory address is >= 27, all of memory cannot be addressed with INDOFF. In these cases, use

INDOFF2 to set the offset while still using INDOFF to set other parameters. Always have INDOFFUPD set to NO_UPDATE in these cases.

- For register accesses, using INDOFF (with INDOFFUPD set to UPDATE) is always more efficient, since it only requires one write.
- **HOST1X_CHANNEL_INDCNT_0:** Indirect register access count used to trigger indirect reads. Holds the number of registers/memory locations that will be read out. Channels should not request more than there is space available in their output FIFO. For indirect frame buffer reads, each channel cannot issue more than NV_HOST1X_MAX_IND_FB_READS at once. The read data must return and be written into the per-channel output FIFO before any additional reads can be issued.
- **HOST1X_CHANNEL_INDDATA_0:** This register, when written, writes to the data to the INDOFFSET in INDOFF. For reads, a REGFNUMEMPTY number of 32-bit values can be read before needing to poll FIFOSTAT again. The per-channel output FIFO (OUTFENTRIES) is readable via this offset. A read of INDDATA will pop an entry off of the per-channel output FIFO.
- **HOST1X_CHANNEL_CMDSWAP_0:** Command swap control. Affects swapping on writes to the PIO region and the frame-buffer buffered memory write region.
- **HOST1X_CHANNEL_FIFOSTAT_0:** CFNUMEMPTY is the number of free slots available in the per-channel command FIFO (needed for PIO or polling for completion of a wait).

12.3.1.2 Channel Command FIFO

The command FIFO is loaded either the CDMA or via PIO access. PIO is mainly used for debug purposes, but it allows a Host1x master to fill the command FIFO with channel commands by writing into “command FIFO” region of the channel's address space. PIO Command FIFO accesses should not be issued while a “Gather” process is busy.

A command FIFO has a set of registers that point to a location in memory where a sequence of channel commands resides. This sequence of commands is generally referred to as a “Push-buffer”.

A Host1x master can write either to DMAPUT or DMASTART register to trigger command fetching by the CDMA. DMASTART indicates where to start fetching, and DMAPUT indicates where to stop. DMAGET is a read-only reference to the present location of the command FIFO; it is incremented when the command is popped from the command FIFO. DMAPUT and DMAGET are relative addresses to DMASTART.

DMAEND provides an upper boundary to prevent the CDMA from fetching illegal addresses; fetching will cease when DMAGET equals DMAEND.

The command FIFO can be halted by writing the DMASTOP field in the Dmactrl register. Dmactrl also provides a mechanism to reset the DMAGET pointer (*Dmagetrst* field) to either 0 or to the value of DMAPUT (Dmainitget field).

12.3.1.3 Channel Commands

Channel commands can be split into two categories:

- **Class commands:** A class command is a mechanism to communicate a class write to a client. The channel must have an active class when processing class commands. The active class is set by a SETCL (SetClass) command.
- **DMA commands:** DMA commands control what is being fetched. They are processed at the top of the command FIFO while class commands are processed at the bottom of the command FIFO. This is because DMA commands change the command stream and must be processed before entering the FIFO.

Offsets in class commands are relative to the active class' base as they are limited to only the methods in the active class.

Channel Commands

- **SETCL (SetClass)** – A channel command that takes a class ID as an argument. SetClass is the sole means to indicate the current virtualization of the channel. It also means that a channel can acquire ownership of a class. Subsequent channel commands are directed towards this class.
- **NONINCR (NonIncrement) –** Takes an offset and count as arguments. There will be <count> data following this class command that will be written to the specified offset. Nonincrement indicates that the offset will not be incremented per data write.
- **INCR (Increment) –** Takes an offset and count as arguments. There will be <count> data following this command that will be written starting at the specified offset. The offset is incremented after each write.
- **MASK (Mask) –** Takes an offset and count as arguments. The number of data to write after the command is equal to the number of set bits in the count. Each data is written to the specified offset plus the next active bit location. For example, a count equal to 0x5, would have 2 data that are written to (offset+0) and (offset+2).
- **IMM (Immediate) –** Takes an offset and 16-bit data as arguments. The 16-bit data is written to the offset.

DMA Commands

- **RESTART (Restart or Jump) –** This command specifies an offset relative to DMASTAT. The next command fetched will be from (DMASTART + offset).
- **GATHER** – Command comprises 2 words and has arguments: offset, insert, type, count, and address. When GATHER is processed, <count> data is fetched from the address and inserted into the command stream. If the <insert> argument is enabled, it will insert either an INCR or NONINCR opcode preceding the fetched data, which is specified by <type>.

DMA commands do not need an active class, but often are processed when an active class exists. In the case of GATHER, if <insert> is enabled, there must be an active class.

12.3.1.4 Channel Control

Channel status and control resides in the synchronous register space of the Host1x address map. These registers are aliased in each channel's space. The details of these registers are as below:

- **CH<0..n>_STATUS:** Status includes client ownership and whether the channel is blocked or not (n is the number of channels – 1).
- **CH_TEARDOWN:** Using this register each channel can be reset, which is referred to as a teardown. This means all channel state is cleared and all client and class ownership is relinquished.
- **MOD_TEARDOWN:** Similar to channel teardown, there exists a mechanism to reset modules. Setting this register can clears all state associated with a given client/module.
- **<client>_STATUS** : This register indicates the client's currently active class.

12.3.2 Host1x Class

Host1x has its own class that can be executed from the command FIFO. It is comprised of methods to access client registers as well as methods to control channel flow.

12.3.2.1 INDOFF

The following Host1x methods operate identically to the channel registers used for indirect access. Refer to the Host Channel Registers and Indirect Register Access sections.

- INDOFF
- INDOFF2
- INDCTRL

- INDDATA

INDCNT is absent from the list above. In Host1x master-initiated indirect register reads, INDCNT is written to trigger the read. Conversely, reads from the command FIFO are trigged by a write to INDDATA. Reads issued from the command FIFO are returned to the channel's register return FIFO (return_FIFO).

12.3.2.2 DELAY_USEC

Delay_Usec stalls the command FIFO for the number of microseconds specified. This command has no impact on indirect register accesses if **not** initiated from the command FIFO. The microsecond period is calculated based on setting in the Usec_Clk register. The Usec_Clk register is programmed on the basis of Host1x clock for example if host clock is 250 MHz, then this register should be programmed to a value of 250.

12.3.2.3 TICKCOUNT

Tickcount_Hi, Tickcount_Lo, and Tickctrl can also be controlled from the command FIFO. Their operation is identical whether controlled from the command FIFO or a Host1x master.

12.3.2.4 INCR_SYNCPT

All Host1x modules, including the Host1x itself, implement the Incr_Syncpt class.

Incr_Syncpt <Cond> <Indx>

For the Host1x, this method immediately increments Syncpt[Indx] irrespective of the cond.

12.3.2.5 WAIT_SYNCPT

Wait on syncpt – the command dispatch will stall until the syncpt counter pointed by the index field reaches the threshold value specified in the Thresh field:-

Wait_Syncpt <Indx> <Thresh>

The channel will wait until the following is true:

Syncpt[Indx][15:0] >= Thresh[15:0]

where the “>=” takes into account wrapping (see the “Sync Points (SYNCPTs)” section for more information on wrapping). More specifically, the channel will wait until:

((Syncpt[indx] – thresh) & (1 << 15)) != 0

12.3.2.6 WAIT_SYNCPT_BASE

This method uses Syncpt Base registers to calculate threshold value for channel wait operation. The syncpt index, base index and also optional offset will be a part of Wait_Syncpt_Base command:-

Wait_Syncpt_Base <Indx> <Base_Indx> <Offset>

The channel will wait until the following is true:

Syncpt[Indx][15:0] >= (Syncpt_Base[Base_Indx] + Offset)[15:0]

Where the “>=” takes into account wrapping. See the “Sync Points (SYNCPTs)” section for more information on wrapping.

12.3.2.7 WAIT_SYNCPT_INCR

Wait until syncpt increments:

Wait_Syncpt_Incr <Indx>

The channel will stall until Syncpt[Indx] is incremented. Note that this wait method is not recommended.

12.3.2.8 LOAD_SYNCPT_BASE

Load a new value into the Syncpt_Base register:

load_syncpt_base <base_indx> <value>

Syncpt_Base[Base_Indx] will be loaded with <Value>.

12.3.2.9 INCR_SYNCPT_BASE

Add an offset to the value in the Syncpt_Base register:

Incr_Syncpt_Base <Base_Indx> <Offset>

The following operation will be done:

Syncpt_Base[Base_Indx] += Offset

12.3.2.10 STALLCOUNT

STALLCOUNT_HI, STALLCOUNT_LO, and STALLCTRL are methods to control “stall counters” through the command FIFO. Details of stall counters are given later in this document.

12.3.2.11 XFERCOUNT

Channel_Xfer_Hi, Channel_Xfer_Lo, and Xferctrl are methods to control “xfer counters” through the command FIFO.

12.3.2.12 32-Bit Sync Point Comparison Methods

Five methods are used for 32-bit sync point comparison. Software should send these commands using the following format:

LOAD_SYNCPT_PAYLOAD_32, <other_command>

where <other_command> is one of the following::

- WAIT_SYNCPT_32
- WAIT_SYNCPT_BASE_32 ,
- LOAD_SYNCPT_BASE_32,
- INCR_SYNCPT_BASE_32

LOAD_SYNCPT_PAYLOAD_32 <Payload(32)>

This method loads a 32-bit value into the corresponding channel's Channel_Syncpt_Payload register:

Channel_Syncpt_Payload[31:0]= <Payload(32)>

WAIT_SYNCPT_32 <Indx(8)>

This method stalls the current channel until following condition is true:-

(SYNCPT [<indx>][31:0] - PAYLOAD[31:0]) & 0x80000000) == 0

Here the Payload value is taken from Channel_Syncpt_Payload of the current channel.

This is essentially a wrapping stall until ((SYNCPT [<indx>][31:0] >= PAYLOAD[31:0])

WAIT_SYNCPT_BASE_32 <Indx(8)> <Base_Indx(8)> :

This method stalls the current channel until following condition is true:-

((Syncpt [<Indx>][31:0] - (Payload[31:0] + Syncpt_Base[<Base_Indx>][31:0]) & 0x80000000) == 0

Here the Payload value is taken from Channel_Syncpt_Payload register of the current channel.

This is essentially a wrapping:-

stall until ((Syncpt [<Indx>][31:0] >= (Payload[31:0] + Syncpt_Base[<Base_Indx>][31:0]))

LOAD_SYNCPT_BASE_32 <Base_Indx(8)>

This method copies the value from the current channel's Channel_Syncpt_Payload register into the Syncpt_Base register specified through the Base_Indx field:-

Syncpt_Base[<Base_Indx>][31:0] = Channel_Syncpt_Payload[31:0]

INCR_SYNCPT_BASE_32 <Base_Indx(8)>

This method adds the value of the current channel's Channel_Syncpt_Payload register into the Syncpt_Base register specified through Base_Indx:-

Syncpt_Base[<Base_Indx>][31:0] += Channel_Syncpt_Payload[31:0]

12.3.3 Context Switching

Context management of Host1x 1x-based modules will be completely under software control (there will be no automatic context switching done by hardware). This mean a module will never produce a context switch interrupt, it will always operate in auto-acknowledge mode.

12.3.4 Behavior of SetClass

The SetClass does not mean implicit acquisition of a module's ownership. It is possible to send commands simultaneously from more than one channel to the same module. If exclusive ownership of a module is required for software correctness, then acquire_mlock and release_mlock opcodes should be used, which acquire and release the MLOCKn semaphore bits. An acquire_mlock that fails will cause that channel's commands to stall until it wins subsequent MLOCK arbitration (arbitrations happen with each release_mlock).

Here is the summary of SetClass behavior:-

- SetClass simply instructs the channel hardware which module and context to use for the following commands.
- If multiple channels write simultaneously to the same class ID, and no MLOCKs are used, then the actual commands are interleaved in round-robin fashion (one command per channel).
- If multiple channels write simultaneously to different class IDs in the same hardware module (and no MLOCKs are used), then the actual commands are interleaved, but in blocks of N-cycle bursts (each burst of commands must be preceded with a CTXSW to the module giving the new context ID).

  N in this case is programmable number configured through Ctxsw_Timeout_Cfg register.

12.3.5 Sync Points (SYNCPTs)

A Sync point is a mechanism to synchronize between software and Host1x clients and also in between Host1x clients. This is implemented as 32-bit counters which are incremented by 1 whenever some predestinated condition (or event) occurs. When a counter reaches its maximum value, it wraps back to zero on the next increment.

Tegra K1 processors have 192 sync points. Sync points are not permanently associated with a channel; sync point allocation is done by software during initialization time.

There are two basic ways for sync point increments: -

- When a CPU writes an index to the Host1x's "syncpt_cpu_incr" register.
- When a Host1x client has received an "incr_syncpt" method and the condition specified by this method has become true.

Synchronization using sync points can be done in the following ways:

- A CPU can be interrupted when a sync point reaches a pre-specified value.
- A Host1x channel can have "wait" commands so that channel will wait for a pre-specified sync point value.

Sync points are normally not reset, but can wrap -- the comparison takes into account the possibility of wrapping.

Note:	Sync point wrapping works only if (Syncpt value - Syncpt Threshold) <= $2^{(syncpt\_width-1)}$ For 16-bit syncpt comparisons, the difference should be less than or equal to 32768. Software must take care of wrapping issues. For 32-bit comparisons, the difference should be less than or equal to 2,147,483,648, Because of the large value, software will not see any wrapping issues.

All Host1x clients (e.g., VI) implement the following increment syncpt method:

Incr_Syncpt <Condition> <Index>

The Host1x client would receive the "Incr_Syncpt" method and store the index for each condition. Whenever the "condition" event occurs, the client would return the index back to Host1x.

12.3.5.1 Client Model for Syncpt Behavior

Software Programming Model

The basic programming model that software will follow is:

Each module will be programmed to do a unit of work (an operation) by Host1x using CDMA and push buffers.

Examples of an operation include:-

- BLT (VIC)
- Draw a set of triangles (GPU)
- Encode a single frame (MSENC).

If nothing else is programmed, then module will go idle until Host1x sends commands to start another operation (no continuous mode).To do its operation, a module reads data from memory and writes the results to memory. Modules interact with each other using memory buffers: one module is the producer of data, and another is the consumer of that data.

There are exceptions to this model which we will discuss in detail later, but should be mentioned here. The exceptions include VI and DISPLAY which work in continuous mode (they continue to process data without the need of additional Host1x programming).

Basic Synchronization

There are two basic needs for synchronization: management of memory buffers and timing of control register writes.

Memory buffers used to pass data from one module to the next use a producer/consumer model with circular buffers. To prevent buffer underflow and overflow, synchronization needs to be done in both directions:

- The consumer cannot read until producer is done writing (and the writes are committed to memory).

- The producer cannot reuse an output buffer (i.e., write to buffer) until the consumer is done reading the buffer.

Thus, the synchronization events required for memory buffers are:

- The module has completed all reads from the buffer.
- The module has completed all writes to the buffer (and they are committed by the memory controller).

To understand the requirements for timing the writes to control registers, a typical sequence is provided below:

1. reg wr for operation A
2. reg wr for operation A
3. reg wr (trigger) for operation A
4. reg wr for operation B
5. reg wr for operation B
6. reg wr (trigger) for operation B

If no WAIT method is placed between the trigger for A and the first register write for B, then in the worst case, corruption of operation A may occur because the new value of the control register is used before operation A is done. For modules that protect against this corruption, there is still the undesirable behavior of the module delaying the register write and subsequently causing back pressure on the Host1x write bus. If we wish to allow direct register reads to be used by ISRs they will happen asynchronously to the channel command writes, so one must ensure that the Host1x bus does not stall for significant periods of time.

For synchronizing writes to control register, a safe time to start writing register for the next operation is defined to be when:

- No corruption will occur for previous operations.
- No stall of the HWRBUS bus will result.

12.3.5.2 Standard Set of Incr_Syncpt Conditions

The following values of the “incr_syncpt cond” field are predefined:

- **0** (Immediate): Return indx to Host1x immediately (used by software push-buffer allocation and helpful for debug).
- **1** (Op_Done): Return indx to Host1x when all previously triggered operations have completed and their writes to memory are committed.
- **2** (Rd_Done): Return indx to Host1x when all previously triggered operations have completed their reads from memory.
- **3** (Reg_Wr_Safe): Return indx to Host1x when it is safe to program registers for the next operation. Safe means no corruption to previous operations and no stalling of Host1x write bus will occur.

There are special cases which would require use of additional condition values. Some modules have multiple read buffers condition = 2 (all reads done) would mean all reads to all buffers done, but it might be useful to software to know when reads are done to a specific input buffer. For some modules, e.g., VI, there are different safe times to update different sets of registers, so condition = 3 will need several variations (one for each "safe time"). If a module can have two operations happening at one time (as in VI), then special considerations will need to be made for these cases: either two separate Incr_Syncpt methods or one Incr_Syncpt method with many special conditions.

For some modules, one condition can replace another if the two conditions always happen within a short period of time of each other. Then the condition that always happens last can be used for both.

12.3.5.3 Continuous Mode — Display and VI

For each display, software wants a free-running syncpt increment on every display “Vsync”. The most appropriate implementation of this is to have an additional register which has “Enable” and “Indx” fields to control the increment of syncpt on every “Vsync” event. If enable is set, then on every “Vsync”, display would return this “Indx” back to Host1x.

A similar situation exists for VI, where besides a set of conditions for the “Incr_Syncpt” method, it would also need a register with “Enable” and “Indx” which would control the incrementing of a syncpt on every camera “Vsync”.

Host1x clients will implement the logic associated with these registers. When this continuous mode is enabled, the client should set a pending bit for when the condition occurs. When the pending bit is set, the client will arbitrate for the hrd_<client>2host1x bus and if selected it will send “Indx” tagged with “Syncpt type” on hrd_ bus. After successfully sending the indx on hrd bus, the client will clear the pending bit.

One of the issues for continuous operation of VI is that not all modes of operation allow the consumer of their output buffers to signal backpressure when the consumer has fallen behind in reading. To alleviate this problem, VI has registers which, when written to, signal that the reading of one buffer has completed.

See Table 32 for a list of the Host1x clients.

12.3.5.4 Allowing Multiple Pending INCR_SYNCPTS

Clients can have multiple pending syncpts which are queued into the client’s syncpt FIFOs, which are dedicated ones for each syncpt conditions.

The behavior of these syncpt FIFOs is controlled by an “Incr_Syncpt_Cntrl” register in client’s register space:

INCR_SYNCPT_CNTRL< No_Stall>< Soft_Reset>

- No_Stall:

  If this bit is 1 and an Incr_Syncpt method is received but the syncpt FIFO for that condition is full, then this method will be dropped and the Incr_Syncpt_Error[Cond] bit will be set.

  If this bit is 0, then instead of dropping next method in case of a syncpt FIFO full condition, the client will just stall the host interface.

- Soft_Reset:

  if Soft_Reset is set, then all internal state of the client syncpt block will be reset. To do a soft reset, first set Soft_Reset of all Host1x clients affected, then clear all Soft_Resets.

INCR_SYNCPT_ERROR<Cond_Status>

This register stores error status in case of above mentioned syncpt FIFO full condition.

12.3.5.5 32-bit Sync Point Comparison

In Tegra 3 and previous chips, sync point counters were implemented as 32-bit registers but comparison takes into account 16-bit value only. In Tegra K1 devices, sync point comparison is extended to 32 bits. To support this feature, Host1x has the following changes:-

Host1x 32-Bit Registers

The following registers are extended to 32 bits:

- Syncpoint base register [Host1x 1x_Sync_Syncpt_Base]
- Syncpt Threshold register [Host1x 1x_Sync_Syncpt_Int_Thresh]

The following 32-bit register stores the payload value required in syncpt methods:

- Channel_Syncpt_Payload[31:0]

Host1x Method Changes

To support 32-bit syncpt comparisons, new Host1x methods have been introduced and are described later in this document.

12.4 Unit Description

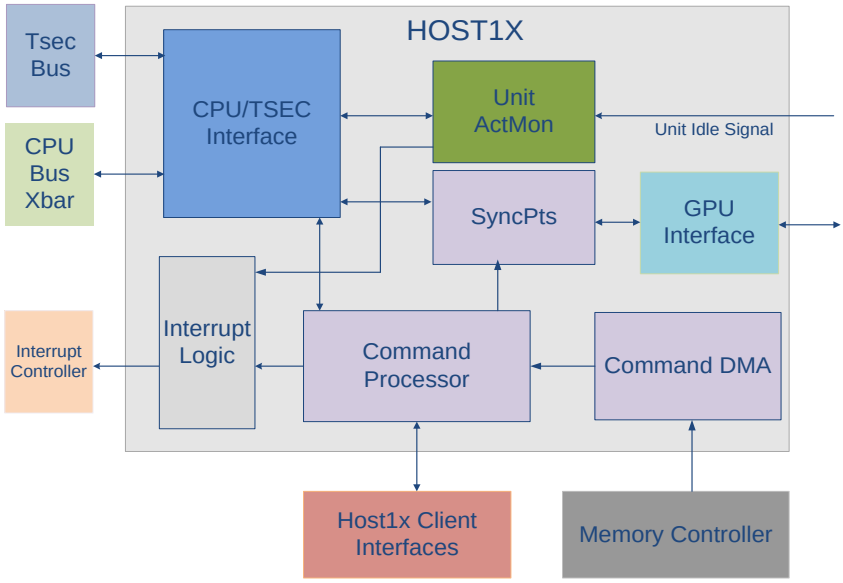
Host1x has a master client interface with the CPU Xbar bus and also a TSEC unit to receive CPU/TSEC read/write commands. The incoming commands are either processed inside Host1x if they are Host1x specific or routed to clients.

Host1x has a point-to-point interface with its clients using HWR/HRD buses, which are used for sending requests and accepting responses from clients.

There is a memory controller interface to fetch Push-Buffer commands from memory through the “Command DMA” unit. The incoming Push-Buffer commands are processed inside “Command Processor” and afterwards either consumed inside Host1x or it will generate tractions to client interface. There is an internal “syncpt” unit which is used to synchronize between Host1x clients and also between software.

There is a “Unit ActMon” block to monitor activities of Host1x clients. The ActMon statistics are used by software for power management.

Figure 17: Host1x Top-Level Block Diagram

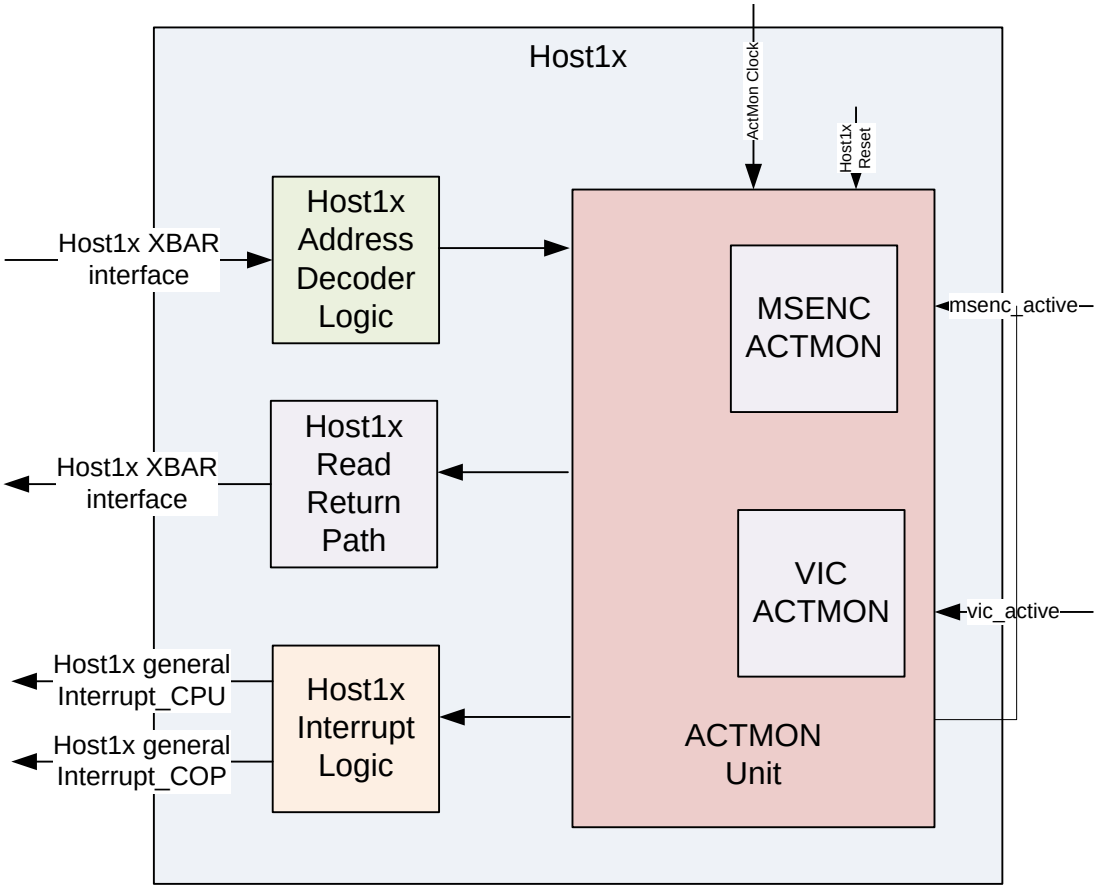


12.4.1 Activity Monitor

The Host1x unit implements 32-bit activity monitor (ACTMON) counters to monitor the idleness of the MSENC/VIC units.

The MSENC/VIC units send active information to Host1x through the same “active” signal (which is a level signal). The ACTMON block provides averaging, watermark detection, and interrupt functionality similar to the central activity monitors. The Host1x driver manages these monitors.

Figure 18: ACTMON Interface



The clock for the ACTMON unit is a low-frequency (10 MHz ~ 50 MHz), non gateable clock.

There is 1 interrupt status bit (per MSENC and VIC ACTMON interrupt) in Host1x, which is set by the actmon_intr_set_msenc and actmon_intr_set_vic signals and cleared by software. Also there are corresponding interrupt mask bits for these interrupt lines to enable/disable interrupts.

The ACTMON register accesses are handled by the ACTMON itself. The Host1x will forward the access control/address/data signal to the ACTMON unit.

The ACTMON registers are a part of Host1x register space.

12.4.2 Filtering Out Kernel Commands from the Gather Buffer

This feature maintains user mode driver versus kernel driver protection. The host1x kernel driver sets up mode bits, which need to be protected from the user mode driver, so that driver cannot directly break the kernel.

Host1x checks the incoming commands from the gather buffer. If it contains the SetClass kernel mode command, then Host1x should drop these commands and raise an “Invalid Gbuffer cmd” interrupt to software. Software will read the interrupt status register to detect the channel that received the invalid gbuffer command. That particular channel needs to be restarted through the “channel teardown” mechanism.

- HOST1X_SYNC_HINTSTATUS_EXT1_0[0:11]

  Bit (i) :-ch(i)_Invalid_gbuffer_cmd_int_status

  Where i = 0..11

- HOST1X_SYNC_HINTMASK_EXT1_0[0:11]

  Bit (i) :- ch(i)_Invalid_gbuffer_cmd _int_mask

  Where i = 0..11

There might be some cases where a channel can still see setClass, for example, context switching. For those channels this feature must be disabled. This feature is controlled through CHANNELCTRL for each channel:

CHANNELCTRL[2]

- 0: disable filtering of kernel command(setClass).
- 1: enable filtering of kernel command(setClass).

12.4.3 Timeout Mechanism

Host1x has a mechanism to generate an interrupt for unserviced CPU read/write requests, which is controlled through a timeout register (HOST1X 1x_IP_TIMEOUT). Host1x generates a timeout Interrupt, and a violating address is stored in the following registers:

- IP_READ_TIMEOUT_ADDR
- IP_WRITE_TIMEOUT_ADDR

The width of the timeout register is 32 bits to provide a timeout value of ~19 seconds.

The TSEC master client interface also uses this mechanism to generate a timeout interrupt to CPU in case of illegal read/write access. The HINTSTATUS_EXT register has been modified to indicate if the timeout is from native CPU access or TSEC access.

Also in case of a TSEC read timeout, the returned response packet will have an additional bit (tsec2host1x 1x_iprdata_timeout) to indicate a timeout error.

HINTSTATUS[Timeout_source=bit10] =

- 0 Timeout from TSEC access.
- 1 Timeout from XBAR access.

The current scheme saves only the last timeout address in case of multiple timeout happens before clearing the timeout interrupt by software.

12.4.4 Performance Statistics Counters

Host1x has performance counters for profiling, which can be enabled independent of push-buffer commands. Host1x needs the following statistics per channel:

- Total clocks
- Clocks stalled waiting for syncpt
- Clocks transferring data

12.4.4.1 Tick Counter

Each channel has its own 64-bit tick counter that is incremented on each Host1x clock, given that it is enabled. TICKCTRL enables and disables the tick counter.

The 32-bit TICKCOUNT_LO and TICKCOUNT_HI registers can be written to initialize the tick counter. Reads of these registers return the tick count. TICKCOUNT_LO stores the lower 32 bits, and TICKCOUNT_HI stores the upper 32 bits. TICKCOUNT_HI is calculated as follows:

if (TICKCOUNT_LO == 0xFFFF_FFFF && TICKCTRL == enable) TICKCOUNT_HI++;

12.4.4.2 Channel Stall Counter

The 64-bit CHANNEL_STALL counter per channel, when enabled, is incremented on each clock cycle if that particular channel is waiting for syncpt.

The following registers control this counter:

- STALLCTRL: This 1-bit register enables/disables the CHANNEL_STALL counter.
- STALLCOUNT_LO: This 32-bit register initializes the lower 32 bits of the CHANNEL_STALL counter.
- STALLCOUNT_HI: This 32-bit register initializes the upper 32 bits of the CHANNEL_STALL counter.

The following Host1x class methods are used to program this counter:

- STALLCOUNT_HI: This method initializes the high 32 bits of the tick count value in the CHANNEL_STALL counter.
- STALLCOUNT_LO: This method initializes the low 32 bits of the tick count value in the CHANNEL_STALL counter.
- STALLCTRL: This method enables/disables the CHANNEL_STALL counter.

12.4.4.3 Channel Xfer Counter

There is a 64-bit CHANNEL_XFER counter per channel to measure the data transfer interval per channel. This counter is incremented at each clock cycle, if the channel is not idle (busy in fetching channel commands).

The following registers are used to initialize this counter:

- XFERCTRL: This 1-bit register enables/disables the CHANNEL_XFER counter.
- CHANNEL_XFER_LO[31:0]: This 32-bit register initializes the lower 32 bits of the CHANNEL_XFER counter.
- CHANNEL_XFER_HI[31:0]: This 32-bit register initializes the lower 32 bits of the CHANNEL_XFER counter.

The following Host1x class methods are used to program this counter:

- CHANNEL_XFER_HI: This method initializes the high 32 bits of the tick count value in the CHANNEL_XFER counter.
- CHANNEL_XFER_LO: This method initializes the low 32 bits of the tick count value in the CHANNEL_XFER counter.
- XFERCTRL: This method enables/disables the CHANNEL_XFER counter.

Note: Counter operations are identical irrespective of whether they are controlled through registers or other methods.

12.4.5 TZ Non-Secure Bit for the Display Engine

To support programming of a Host1x client (display engines) in TZ (Trust Zone) mode through direct register access, the Host1x needs to transfer a TZ non-secure bit from the CPU to the display and VI engines.

The following table describes the behavior of TZ non-secure signal to clients.

Table 31: Behavior of TZ Non-Secure Signal

Access-Mode	CPU TZ NS bit	TZ Non-Secure Bit to Client
Direct MMIO access(read/write)	0	0
Direct MMIO access(read/write)	1	1

Access-Mode	CPU TZ NS bit	TZ Non-Secure Bit to Client
Indirect MMIO access(read/write)[1]	0	X
Indirect MMIO access(read/write)	1	1
Push-buffer access(write)(DMA/PIO)	X	1
Push-buffer access(Indirect reads)(DMA/PIO)	X	1

12.4.6 Interrupts

Host1x has following types of interrupts:

12.4.6.1 Host1x Client Interrupt

Host1x is the collection point for all of its clients interrupts. Also it has control over those using status and mask registers with fields for each client. These client interrupts are forwarded to the central interrupt controller unit, individually through per client interrupt lines.

12.4.6.2 Host1x Interrupts

Host1x -specific interrupts are specified through the HOST1X_SYNC_HINTSTATUS_0 register while additional interrupts have been specified through the HOST1X_SYNC_HINTSTATUS_EXT_0 register (see Registers below). Each Host1x interrupt also has a corresponding mask. An interrupt is cleared by writing a 1 to the corresponding bit.

12.4.6.3 Host1x Syncpt Interrupts

Host1x can generate a syncpt interrupt upon reaching a threshold value by syncpt registers. It can also generate an interrupt to either CPU (referred to as CPU0 below) or AVP (referred to as CPU1 below).

An interrupt is routed to cpuN when:

> (SYNCPT[indx] >= SYNCPT_INT_THRESH[indx]) && (SYNCPT_THRESH_INT_MASK[indx] == cpuN)
>
> The comparison takes wrapping into account. See the “Sync Points (SYNCPTs)” section for more information on wrapping.

The status is sticky and is PENDING until cleared. Write 1's to clear.

12.4.7 Indirect Register Access

Host1x provides indirect register access to all its clients. An indirect register access involves multiple steps and requires a channel for the read return data. All registers required for indirect accesses reside in a channel.

Writes involve two Host1x register writes in the owning channel's space; the first write indicates the address (INDOFF, or INDOFF2 and INDCTRL), and the second indicates the data (INDDATA).

Reads involve two direct Host1x writes, the first indicates the address (again INDOFF, or INDOFF2 and INDCTRL), and the second indicates the number of reads (INDCNT). It also requires an indeterminable number of direct Host1x reads to the read return FIFO status register followed by a read to the read return FIFO.

- INDOFF
- INDOFF2
- INDCTRL

[1] This mode is not supported for “secure access”, The CPU has to use only “direct access” in “secure mode”.

- INDDATA
- INDCNT

Pitfalls — Indirect accesses require a software mutex to ensure that no other software threads access the indirect registers while issuing the access (since the series of accesses is non-atomic). Indirect reads can only issue counts less than the read return FIFO size to prevent a deadlock – too many read requests can block popping of the read return FIFO. This restriction may be even tighter. All module accesses must also be tied to a channel.

In case an indirect write is sent, software must check whether the previous indirect write is through using an immediate sync increment command to the client.

12.4.8 Direct Register Access

All clients under Host1x have 256KB of address space, which originates from the space specified by indirect offset register (INDOFF). Host1x's client address map is dictated by its client's module IDs; these IDs are used in INDOFF to specify the target module of the register access. A client's address in the Host1x space is calculated as:

address = direct_access_base_address (0x54000000) + (module_id << 18) + 4 * register_offset

Only one pending read per interface can exist. Reads are returned from the client when the client is ready. There are no latency restrictions.

Indirect versus Direct — a pitfall of direct addressing is variable latency of Host1x's clients. While this has no impact on writes, the CPU is blocked until the read returns rendering the CPU idle during that time.

12.4.9 Host Clients

The following table lists the clients that are accessed via Host1x with their module (client) ID, which is used to determine addresses for direct addressing.

Table 32: Host1x Clients

Client	ID
Host1x	0x0
VI	0x2
Display A	0x8
Display B	0x9
HDMI	0xa
DSI	0xc
VIC	0xd
CSI	0xf
DSIB	0x10
MSENC	0x13
TSEC	0x14
SOR	0x15
DPAUX	0x17
ISP	0x18
ISPB	0x1a

12.4.10 Class IDs

The following table lists the class IDs.

Class	ID
NV_HOST1X_CLASS_ID	0x01
NV_VIDEO_ENCODE_MSENC_CLASS_ID	0x21
NV_VIDEO_STREAMING_VI_CLASS_ID	0x30
NV_VIDEO_STREAMING_ISP_CLASS_ID	0x32
NV_VIDEO_STREAMING_ISPB_CLASS_ID	0x34
NV_GRAPHICS_VIC_CLASS_ID	0x5D
NV_DISPLAY_CLASS_ID	0x70
NV_DISPLAYB_CLASS_ID	0x71
NV_HDMI_CLASS_ID	0x77
NV_DISPLAY_DSI_CLASS_ID	0x79
NV_DISPLAY_DSIB_CLASS_ID	0x7A
NV_TSEC_CLASS_ID	0xE0
NV_SOR_CLASS_ID	0x7B
NV_DPAUX_CLASS_ID	0x7D

12.4.11 Host Address Space

Each channel is allocated 16KB of space, and they are aligned contiguously at the top of the Host's address space. A channel's space consists of channel registers, the command FIFO, and an aliased frame buffer region, which are unique per channel. The rest of the space consists of asynchronous, synchronous and read DMA registers, which are aliased in each channel – there exists only one copy.

12.4.11.1 Channel Map

The following table gives offsets from the channel base address for different sets of registers available to each channel.

Table 33: Host1x Channel Map

0x0000	Channel Registers	
0x0600	RDMA Registers	
0x0800	Command FIFO	
0x1000	Frame Buffer	
0x2000-0x20FF	Unit ACTMON Registers	ACTMON1(MSENC) = 0x2000 – 0x203F
		ACTMON2(VIC) =0x2040 – 0x207F
		ACTMONRSVD =0x2080 – 0x20FF
0x2100*	Synchronous Registers	
0x3FFF	Bottom	

* 0x2100 is the synchronous registers base (HOST1X_CHANNEL_SYNC_REG_BASE)

12.4.11.2 Host1x Map

The following table shows the Host1x addresses for the CPU. Channels start at the lowest addresses and are aligned contiguously. Direct access to Host clients starts at 5400_0000, but this is configurable through a BAR register.

Table 34: Host1x Map

Address Range	Channel
5000_0000-5000_3FFF	Channel 0 (16KB)
5000_4000-5000_7FFF	Channel 1 (16KB)
5000_8000-5000_BFFF	Channel 2 (16KB)
5000_E000-5001_2FFF	Channel 3 (16KB)
5001_3000-5001_6FFF	Channel 4 (16KB)
5001_7000-5001_AFFF	Channel 5 (16KB)
5001_B000-5001_BFFF	Channel 6 (16KB)
5001_C000-5001_FFFF	Channel 7 (16KB)
5002_0000-5002_3FFF	Channel 8 (16KB)
5002_4000-5002_7FFF	Channel 9 (16KB)
5002_8000-5002_BFFF	Channel 10 (16KB)
5002_C000-5002_FFFF	Channel 11 (16KB)

12.5 Performance

12.5.1 Key Use Cases

Command throughput must be sufficient when all channels are active, the CDMA is active, and spooling is active.

12.5.2 Latency Targets

Direct read latency must be under 1 ms. Clients that cannot guarantee that a read will return in under 1 ms must not be read directly but indirectly.

12.5.3 Bandwidth Targets

As Host1x unit is used only for sending control commands to clients and usually at the frame interval.

As these control commands are short (~200 to 300 words maximum), it is expected that the bandwidth requirement on each client should be less than 1MB/s, whereas the maximum available bandwidth per client on the HWR bus will be (206*4B)/12 =68MB/s at 206 MHz Host1x clock. The minimum bandwidth on the HWR bus per client is 33.34 MB/s at 100 MHz clock which should easily meet the client requirement.

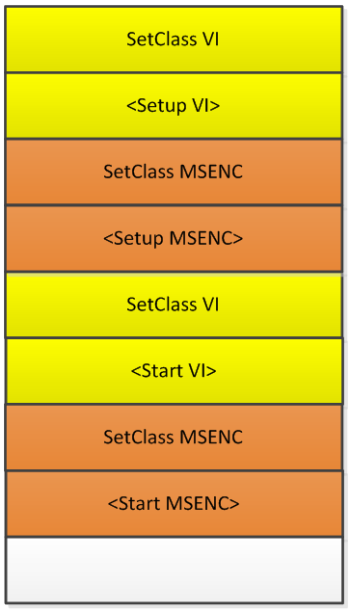
12.6 Host1x Programming Model

The Host programming models resides on top of the concept of channels. A channel provides the means for software to supply an ordered sequence of commands to one or more classes. There are no restrictions on how many classes a channel may own or how often it can switch between classes, but it can only operate on one class at a time (known as the working class) and one command at a time. A stalled channel does not allow any commands to proceed from any class – there is a

strict ordering of commands. A channel starts processing commands simply when commands are present, either supplied by the CPU or gathered from memory. A channel can be stopped explicitly by a CPU write to channel state.

A single channel owning multiple classes and clients:

Figure 19: Single Channel Example



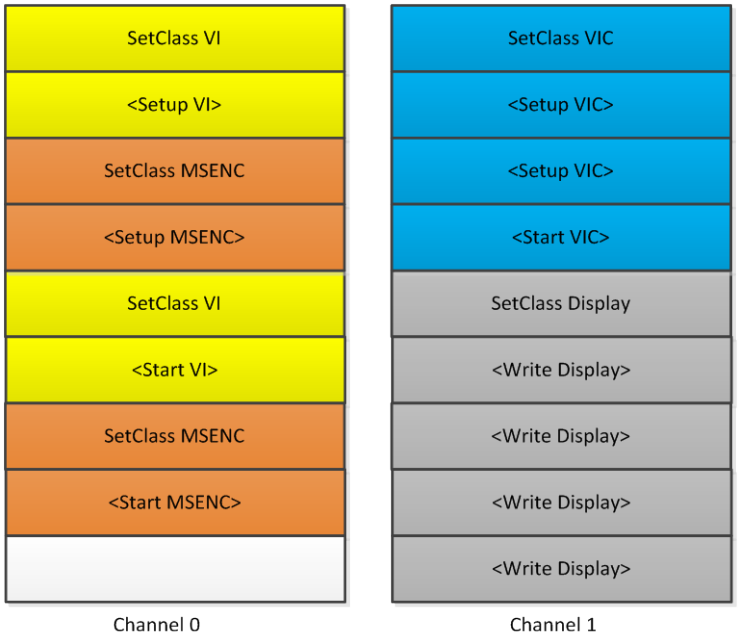
12.6.1 Concurrency

Channels operate in parallel and are unhindered by one another except in two specific cases:

- Synchronization points
- Class contention

The following figure depicts such concurrency.

Figure 20: Channel Concurrency Example



In the example above, no channel is blocked by another at any time so both can work concurrently. Care must be taken when assigning clients to channels. Channel concurrency as well as channel throughput is crucial for performance. For example, a client that is event-driven with a low latency requirement should exist in its own channel. Conversely, two clients that must be steadily supplied with commands could possibly coexist in the same channel given that their synchronization points are not conflicting (if they have any real synchronization points at all). Clients that operate sequentially on the same piece of data can easily reside in the same channel.

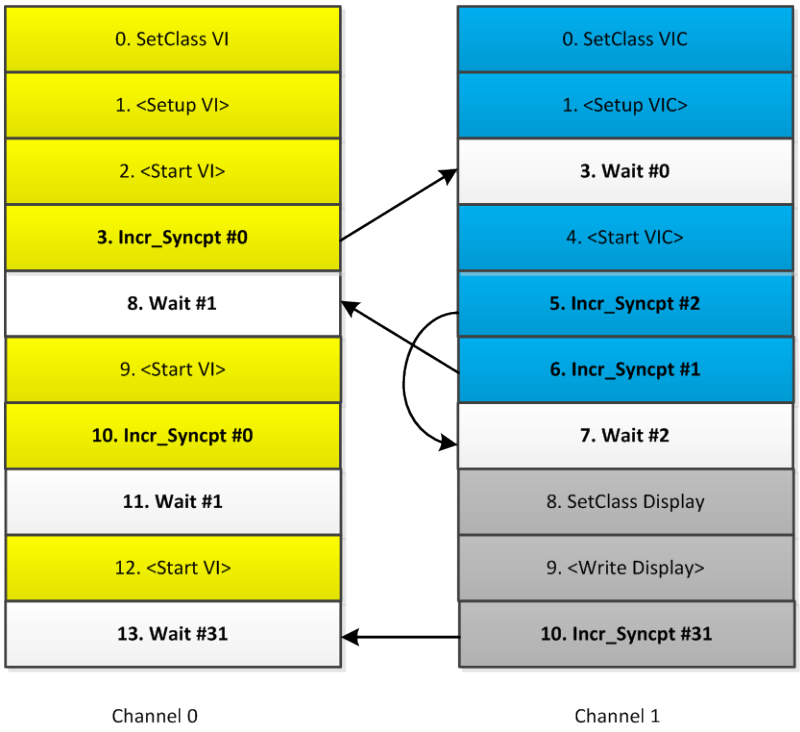
Clients with low-latency requirements like described in the first example above should possibly be moved to a PIO programming model and away from a channel-based model.

12.6.2 Synchronization

There are 192 total synchronization points (syncpts). Similar to semaphores, syncpt methods are issued to clients, mentioning syncpt counter index and returning condition. Any channel can be made to wait on a syncpt value of a particular syncpt index through Host1x “wait” method. Once the syncpt method is received by clients, based on syncpt condition, they will return a syncpt index back to Host1x. On receiving a syncpt index from the client, Host1x will increment that particular syncpt counter.

The following simple example below details how syncpts and waits allow for synchronization. The arrows simply indicate channel dependencies, which channel is waiting for syncpt increment from other channel.

Figure 21: Channel Synchronization Example



12.6.3 Progress Status

Channel progress is monitored through two means – GET and syncpts. GET is a channel state that indicates the address of the last command that has executed in the channel. GET is the same as DMAGET.

Syncpts are counter registers that are incremented when specified events occur (e.g., after the completion of each operation done by a module). The 32-bit syncpt values typically are monotonically increasing and can be used for in channel waits and out-of-channel interrupts.

The GET and syncpt registers can also be read directly (out-of-band) by the CPU.

12.6.4 Syncpt Base Register Use Case

The syncpt base registers are modified through LOAD_SYNCPT_BASE_32 or INCR_SYNCPT_BASE_32 methods and are used with the WAIT_SYNCPT_BASE_32 method.

This mechanism is useful when software wants to wait for a particular syncpt increment, but software does not know the absolute value of the sync point register until later.

The "future-value" of a sync point is the value that the syncpt register contains right after a particular increment has occurred.

Consider this example:

In this example, time increases downward (later (lower) lines are ahead in time).

Software "knows" the syncpointA future-value equals N at this point; software "knows" the baseA future-value equals N at this point.

1. command1
2. increment syncpointA to N+1
3. command2

4. increment syncpointA to N+2

5. command3

6. wait until command1 is done (syncpoint == N+1)

7. command4

8. wait until command2 is done (syncpoint == N+2)

9. command5

If software actually DOES know the value of N then regular WAIT_SYNCPT works fine here. However, the user space software driver (which is creating this list of commands) does not know what the value N is until after these commands (1-9) are all flushed to the kernel space driver.

The kernel space driver receives packets of commands from many different user space processes. When it receives a packet of commands it queues those commands (i.e., writes the Host1x PUT pointer). It also keeps track of all increments that have occurred in all commands that have been queued so far. The "future-value" of each sync point is simply the total number of increments that have occurred since the beginning of time (actually "beginning of time" is really the point in time where the kernel space driver was initialized and reset all the syncpt registers to 0).

So when the kernel space driver queues these commands (1-9), it knows the value N (i.e., the sum of all increments since the beginning of time). It can pass this back to the user space driver. But the user space driver does not know how many increments will be queued before commands 1-9 are queued because any other process can queue commands (including increments) at any time (e.g., after the user space driver has written commands 2 and 4 to the buffer, but before those commands have been queued by the kernel driver).

To solve this, we add a command at the end of each packet that increments the base register the same number of times as the sync point was incremented:

10. baseA += 2

This means that, at the start of any packet of commands, the future value of syncpointA and the future value of baseA are always the same. So command 6 can be:

WAIT_SYNCPT_BASE syncpt=A base=A offset=1

and command 8 can be:

WAIT_SYNCPT_BASE syncpt=A base=A offset=2

Note: Software never writes a new value to the syncpointA or baseA register except when the kernel driver is initialized (e.g., when the system boots). After that only the syncpoint register is incremented and only the base register is added to. All of this depends on all software that creates cooperating command buffers. Software has this policy (e.g., the user space driver always tells the kernel driver how many increments are contained in each packet of commands).

12.6.5 Indirect/Direct Register Addressing Scheme

The indirect addressing scheme works as follows:

- An address is decoded and either routed to a channel register or to a synchronous register
- A write to INDCNT or INDDATA triggers a register access to the register specified in INDOFF
- A transaction is created and pushed into the REGF FIFO. The owning channel is specified by which INDCNT or INDDATA is written. The client and offset are specified in INDOFF.
- The transaction is routed to the target client over the HWR bus.

- If the access is a write, the transaction is complete. In the case of a read, the processor polls the channel's read return FIFO status (FIFOSTAT), waiting for a nonempty FIFO.
- The client returns the read to the channel's return FIFO.
- The processor reads INDDATA register of that particular channel to pops data from the return FIFO.

Direct addressing uses the same flow with some adjustments:

- An address is decoded and either routed to a channel register, a synchronous register, or a client
- This step is nonexistent; direct accesses do not require a trigger.
- In the case of a client decode, a transaction is created and pushed into a FIFO called REGF. The owning channel is specified as CPU_READ_RETURN_TAG. The client and offset are indicated by the address: the client is created by shifting the address down by 18; the offset is simply the lower 18 bits.
- The transaction is routed to the target client over the HWR bus. If the access is a write, the transaction is complete.
- In case of a read, client returns the read data and channel ID in the returned packet indicates its destination, be it interface or a channel's return FIFO. If the channel is 0-8, it is an indirect read and returned to the indicated channel. If the channel matches CPU_READ_RETURN_TAG(0xf), the data is returned to the IP interface.

Currently, there is support for only one pending read per interface.

12.7 Host Channel Opcodes

This section provides the format of opcodes that can be sent through the command FIFO.

HCFCMD

Generic command FIFO packet (contains fields common to all opcodes) and is used for initial decode. All command FIFO packets are multiples of 32 bits.

HCFSETCL

The SetClass opcode specifies which class is being referenced (may cause rerouting of subsequent methods/data). In addition to switching classes, the opcode allows some methods to be programmed on the switch similar to an HCFMASK opcode.

HCFINCR

The Incrementing opcode indicates the offset should be incremented for each data that is part of the packet. The count argument indicates how many 32-bit values are following. If channel protect is enabled, the host should prevent a channel switch from occurring at the end of this command packet.

HCFNONINCR

The Non-Incrementing opcode indicates the same offset should be sent for each data that is part of the packet. The count argument indicates how many 32-bit values are following. If channel protect is enabled, the host should prevent a channel switch from occurring at the end of this command packet.

HCFMASK

The Mask opcode, from the starting offset, generates offsets based on where the bits are set in the mask. The host expects the amount of data following to equal the number of bits set. If channel protect is enabled, the host should prevent a channel switch from occurring at the end of this command packet.

HCFIMM

The Immediate opcode indicates the offset and data are contained in the same 32-bit data. Only the lowest 16 bits of data are sent to the module (IMMDATA). The upper 16 bits are zeroed out.

HCFRESTART

The Restart opcode is specific to DMA operation and causes the host to set DMAGET to (ADDRESS << 4), so the next command fetch will be from (DMASTART + DMAGET).

In legacy chips, bits 27:0 were not decoded and assumed to be 0's (allowing only simply wrapping of GET back to the top of the command buffer). ADDRESS can be 0 for compatible RESTARTs or non-zero acting as a JUMP.

Note that the jump address granularity is 16 bytes, since the bottom 4 bits cannot be specified.

HCFGATHER

The Gather opcode allows contiguous portions of memory to be fetched and placed in line with the command stream, replacing the two words of the gather command. It optionally can put an incrementing or non-incrementing opcode in the stream ahead of the gathered data. This allows for the gathered data to be a pure data stream and not be required to have host opcodes inside.

HCFCHDONE

This opcode indicates to the command processor that the current channel is done processing for now and is willing to give up any of its owned modules to other channels that need them.

12.8 Host Channel Registers

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

12.8.1 HOST1X_CHANNEL_FIFOSTAT_0

CFNUMEMPTY is the number of free slots available in the per-channel command. A FIFO is needed for PIO or polling for completion of a wait.

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

Bit	Reset	Description
31	X	INDRDY: Indicates that INDCOUNT==0, so it should be OK to issue another read
28:24	X	OUTFENTRIES: Number of entries available for reading in this channel's output FIFO
20:16	X	REGFNUMEMPTY: Register write/read FIFO free count
12	X	CFGATHER: Indicates whether GATHER is active. If a GATHER command issued via PIO, software must wait for the GATHER to be IDLE before issuing another command. 0 = IDLE 1 = BUSY
11	X	CFEMPTY: Indicates whether the command FIFO is empty or not 0 = NOTEMPTY 1 = EMPTY
10:0	X	CFNUMEMPTY: Command FIFO free count

12.8.2 HOST1X_CHANNEL_INDOFF_0

The INDOFF and INDOFF2 registers (along with INDCNT and INDDATA) are used to indirectly read/write modules outside the host. If AUTOINC is set, INDOFFSET increments by 4 on every access of INDDATA. REGFNUMEMPTY is polled to determine when valid data can be read from INDDATA.

The INDOFF register has limited capability on chips with large memory maps. If the top bit of the memory address is >= 27, all of memory cannot be addressed with INDOFF. In these cases, use INDOFF2 to set the offset while still using INDOFF to set the other parameters. Always have INDOFFUPD set to NO_UPDATE in these cases. For register accesses, using INDOFF (with INDOFFUPD set to UPDATE) is always more efficient, since it only requires one write.

Indirect frame buffer writes are STRONGLY DISCOURAGED. There are better ways to write to memory (direct and through the channel memory map) and there is limited flow control in the host. It is very easy to get into trouble with indirect frame buffer writes.

Offset: 0x4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	AUTOINC: Auto increment of read/write address 0 = DISABLE 1 = ENABLE
30	X	ACCTYPE: Access type: indirect register or indirect frame buffer 0 = REG 1 = FB
29	X	BUF32B: Buffer up 32 bits of register data before sending it. Otherwise, register writes will be sent as soon as they are received. Does not support byte writes in 16-bit host. Does not affect frame buffer writes. 0 = NOBUF 1 = BUF
28:27	X	INDSWAP: Indirect frame buffer access swap control. 00 = No byte swap 01 = 16-bit byte swap ([31:0] -> {[23:16],[31:24],[7:0],[15:8]}) 10 = 32-bit byte swap ([31:0] -> {[7:0],[15:8],[23:16],[31:24]}) 11 = 32-bit word swap ([31:0] -> {[15:8],[7:0],[31:24],[23:16]}) 0 = NONE 1 = BYTE16 2 = BYTE32 3 = WORD32
25:18	X	INDMODID: ACCTYPE=REG: Register module ID 0 = HOST1X 2 = VI 8 = DISPLAY 9 = DISPLAYB 10 = HDMI 11 = TVO 12 = DSI 13 = VIC 16 = DSIB 19 = MSENC 20 = TSEC 21 = SOR 23 = DPAUX 24 = ISP 26 = ISPB
25:2	X	INDOFFSET: ACCTYPE=FB: frame buffer address
17:2	X	INDROFFSET: ACCTYPE=REG: register offset ([15:0])
0	X	INDOFFUPD: Optionally disable the update of INDOFFSET when writing this register 0 = UPDATE 1 = NO_UPDATE

12.8.3 HOST1X_CHANNEL_INDCNT_0

Indirect Register Access Count

Used to trigger indirect reads. Holds the number of registers/memory locations that will be read out. Channels should not request more than there is space available in their output FIFO. Only the protected channel should make liberal use of this feature for speeding up context switching.

For indirect frame buffer reads, each channel cannot issue more than NV_HOST1X_MAX_IND_FB_READS at once. The read data must return and be written into the per-channel output FIFO before any additional reads can be issued.

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	INDCOUNT

12.8.4 HOST1X_CHANNEL_INDDATA_0

This register, when written, writes to the data to the INDOFFSET in INDOFF. For reads, a REGFNUMEMPTY number of 32-bit values can be read before needing to poll FIFOSTAT again. The per-channel output FIFO (OUTFENTRIES) is readable via this offset. A read of INDDATA will pop an entry off of the per-channel output FIFO.

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

Bit	Reset	Description
31:0	X	INDDATA: read or write data

12.8.5 HOST1X_CHANNEL_RAISE_0

The general-purpose channels have DMA and RAISE/REFCOUNT functionality.

Any raise values returned from a client module are converted to vectors and update the per-channel raise register. The RAISE vector is also writable by the CPU. Any bits set in the RAISE field when written will be set in the RAISE register, allowing any pending WAITs to continue.

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

Bit	Reset	Description
31:2	X	RAISE: This channel's RAISE vector

12.8.6 HOST1X_CHANNEL_DMASTART_0

This register triggers a DMA fetch from the frame buffer for this channel, if the Put register does not equal the DMA Get register.

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:2	X	DMASTART: cmdbuf frame buffer offset

12.8.7 HOST1X_CHANNEL_DMAPUT_0

This register triggers a DMA fetch from the frame buffer for this channel, if the PUT register does not equal the GET register. This address is relative to the DMASTART base address. Does not support byte writes. All 4-byte data need to be programmed.

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000xx)

Bit	Reset	Description
31:2	0x0	DMAPUT: cmdbuf frame buffer offset

12.8.8 HOST1X_CHANNEL_DMAGET_0

This register tracks the frame-buffer offset the DMA engine has read up to (incremented as entries are loaded from the channels command buffer into the FIFO). This address is relative to the DMASTART base address.

Offset: 0x1c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:2	X	DMAGET: cmdbuf frame buffer offset

12.8.9 HOST1X_CHANNEL_DMAEND_0

The boundary of illegal addresses (either end of push-buffer or end of physical memory). This is designed to prevent DMA from prefetching illegal addresses. If DMA reaches this address before seeing a RESTART, it will stop. This would be a software error condition.

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

Bit	Reset	Description
31:2	X	DMAEND: cmdbuf frame buffer offset

12.8.10 HOST1X_CHANNEL_DMACTRL_0

DMA Control Register

DMAGETRST: Reset GET pointer to '0'. Useful for cleaning up crashed channels. DMAGET is not updated instantly. Takes 4 cycles between programming of reset and a valid DMAGET.

DMAGETINIT: Reset the GET pointer to the value of DMAPUT when DMAGETRST is asserted.

DMASTOP: Stop DMA from fetching on this channel.

Note: A Command DMA channel needs to be enabled for PIO-gather to work.

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx001)

Bit	Reset	Description
2	0x0	DMAINITGET: Reset GET pointer to the value of DMAPUT when DMAGETRST is asserted. 0 = DISABLE 1 = ENABLE
1	0x0	DMAGETRST: Reset GET pointer to '0'. Useful for cleaning up crashed channels. 0 = DISABLE 1 = ENABLE
0	0x1	DMASTOP: Stop DMA from fetching on this channel. NOTE: a Command DMA channel needs to be enabled for PIO gather to work. 0 = RUN 1 = STOP

12.8.11 HOST1X_CHANNEL_INDOFF2_0

The INDOFF and INDOFF2 registers (along with INDCNT and INDDATA) are used to indirectly read/write modules outside the host. If AUTOINC is set, INDOFFSET increments by 4 on every access of INDDATA. REGFNUMEMPTY is polled to determine when valid data can be read from INDDATA.

The INDOFF register has limited capability on chips with large memory maps. If the top bit of the memory address is >= 27, all of memory cannot be addressed with INDOFF. In these cases, use INDOFF2 to set the offset while still using INDOFF to set the other parameters. Always have INDOFFUPD set to NO_UPDATE in these cases. For register accesses, using INDOFF (with INDOFFUPD set to UPDATE) is always more efficient, since it only requires one write.

Indirect frame-buffer writes are STRONGLY DISCOURAGED. There are better ways to write to memory (direct and through the channel memory map) and there is limited flow control in the host. It is very easy to get into trouble with indirect frame-buffer writes.

Offset: 0x8c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:18	X	INDMODID2: ACCTYPE=REG: register module ID 0 = HOST1X 2 = VI 8 = DISPLAY 9 = DISPLAYB 10 = HDMI 11 = TVO 12 = DSI 13 = VIC 16 = DSIB 19 = MSENC 20 = TSEC 21 = SOR 23 = DPAUX 24 = ISP 26 = ISPB
31:2	X	INDOFFSET2: ACCTYPE=FB: frame buffer address
17:2	X	INDROFFSET2: ACCTYPE=REG: register offset ([15:0])

12.8.12 HOST1X_CHANNEL_TICKCOUNT_HI_0

This register holds the high 32 bits of the tick count value.

Offset: 0x90 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_HI

12.8.13 HOST1X_CHANNEL_TICKCOUNT_LO_0

This register holds the low 32 bits of tick count value.

Offset: 0x94 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_LO

12.8.14 HOST1X_CHANNEL_CHANNELCTRL_0

This register will be used for controlling some channel-related commands including enabling/disabling of the tick counter, including enabling/disabling of Kernel command filtering from the gather buffers.

Offset: 0x98 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0x0)

Bit	Reset	Description
2	DISABLE	KERNEL_FILTER_GBUFFER: Enable setclass command filter for gather buffers 0 = DISABLE 1 = ENABLE
0	DISABLE	ENABLETICKCNT: enable or disable tick counter 0 = DISABLE 1 = ENABLE

12.8.15 HOST1X_CHANNEL_PAYLOAD_0

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

Bit	Reset	Description
31:0	0x0	PAYLOAD

12.8.16 HOST1X_CHANNEL_STALLCTRL_0

Offset: 0xa0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ENABLE_CHANNEL_STALL

12.8.17 HOST1X_CHANNEL_STALLCOUNT_HI_0

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

Bit	Reset	Description
31:0	0x0	STALLCOUNT_HI

12.8.18 HOST1X_CHANNEL_STALLCOUNT_LO_0

Offset: 0xa8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	STALLCOUNT_LO

12.8.19 HOST1X_CHANNEL_XFERCTRL_0

Offset: 0xac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ENABLE_CHANNEL_XFER

12.8.20 HOST1X_CHANNEL_CHANNEL_XFER_HI_0

Offset: 0xb0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	CHANNEL_XFER_HI

12.8.21 HOST1X_CHANNEL_CHANNEL_XFER_LO_0

Offset: 0xb4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	CHANNEL_XFER_LO

12.8.22 HOST1X_CHANNEL_HOST1X_CHANNEL_SPARE_0

Offset: 0xb8 | Read/Write: R/W | Reset: 0x00ff0000 (0b00000000111111110000000000000000)

Bit	Reset	Description
31:16	0xff	CHANNEL_SPARE_HI
15:0	0x0	CHANNEL_SPARE_LO

12.9 Host SYNC Registers

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

12.9.1 HOST1X_SYNC_INTSTATUS_0

INTSTATUS - interrupt status contains the interrupt status for all of the client modules. These status bits are only status. Writing '1' to them will not clear them. Software must clear the interrupt in the appropriate module, which should be reflected here.

Offset: 0x0 | Read/Write: RO | Reset: 0xXXXXXX0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SYNCPT_CPU1_INT: set if SYNCPT_CPU1_INTSTATUS has a pending interrupt 0 = NOT_PENDING 1 = PENDING
30	X	SYNCPT_CPU0_INT: set if SYNCPT_CPU0_INTSTATUS has a pending interrupt 0 = NOT_PENDING 1 = PENDING
26	X	ISPB_INT: 0 = NOT_PENDING 1 = PENDING
24	X	ISP_INT: 0 = NOT_PENDING 1 = PENDING
23	X	DPAUX_INT: 0 = NOT_PENDING 1 = PENDING
21	X	SOR_INT: 0 = NOT_PENDING 1 = PENDING
20	X	TSEC_INT: 0 = NOT_PENDING 1 = PENDING

Bit	Reset	Description
19	X	MSENC_INT: 0 = NOT_PENDING 1 = PENDING
16	X	DSIB_INT: 0 = NOT_PENDING 1 = PENDING
13	X	VIC_INT: 0 = NOT_PENDING 1 = PENDING
12	X	DSI_INT: 0 = NOT_PENDING 1 = PENDING
10	X	HDMI_INT: 0 = NOT_PENDING 1 = PENDING
9	X	DISPLAYB_INT: 0 = NOT_PENDING 1 = PENDING
8	X	DISPLAY_INT: 0 = NOT_PENDING 1 = PENDING
2	X	VI_INT: 0 = NOT_PENDING 1 = PENDING
0	X	HOST_INT: 0 = NOT_PENDING 1 = PENDING

12.9.2 HOST1X_SYNC_INTMASK_0

Contains a master interrupt mask for all interrupt signals. If the interface's MASK_ALL bit is disabled, no interrupts will be triggered on that interface. This applies to only the non-syncpt interrupts.

Offset: 0x4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	CPU1_INT_MASK_ALL: 0 = DISABLE 1 = ENABLE
0	0x0	CPU0_INT_MASK_ALL: 0 = DISABLE 1 = ENABLE

12.9.3 HOST1X_SYNC_INTC0MASK_0

INTC0MASK - Interrupt Mask for CPU0

Contains the interrupt mask bits for all of the client modules. If the INT_MASK is ENABLED for a particular module, the modules INT bit from INTSTATUS contributes to CPU0's interrupt signal.

Offset: 0x8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxx0x00x000xx0xx00x000xxxxx0x0)

Bit	Reset	Description
26	0x0	ISPB_INT_C0MASK: 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
24	0x0	ISP_INT_C0MASK: 0 = DISABLE 1 = ENABLE
23	0x0	DPAUX_INT_C0MASK: 0 = DISABLE 1 = ENABLE
21	0x0	SOR_INT_C0MASK: 0 = DISABLE 1 = ENABLE
20	0x0	TSEC_INT_C0MASK: 0 = DISABLE 1 = ENABLE
19	0x0	MSENC_INT_C0MASK: 0 = DISABLE 1 = ENABLE
16	0x0	DSIB_INT_C0MASK: 0 = DISABLE 1 = ENABLE
13	0x0	VIC_INT_C0MASK: 0 = DISABLE 1 = ENABLE
12	0x0	DSI_INT_C0MASK: 0 = DISABLE 1 = ENABLE
10	0x0	HDMI_INT_C0MASK: 0 = DISABLE 1 = ENABLE
9	0x0	DISPLAYB_INT_C0MASK: 0 = DISABLE 1 = ENABLE
8	0x0	DISPLAY_INT_C0MASK: 0 = DISABLE 1 = ENABLE
2	0x0	VI_INT_C0MASK: 0 = DISABLE 1 = ENABLE
0	0x0	HOST_INT_C0MASK: 0 = DISABLE 1 = ENABLE

12.9.4 HOST1X_SYNC_INTC1MASK_0

INTC1MASK - Interrupt Mask for CPU1

Contains the interrupt mask bits for all of the client modules. If the INT_MASK is ENABLED for a particular module, the modules INT bit from INTSTATUS contributes to the CPU1's interrupt signal.

Offset: 0xc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxx0x00x000xx0xx00x000xxxxx0x0)

Bit	Reset	Description
26	0x0	ISPB_INT_C1MASK: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
24	0x0	ISP_INT_C1MASK: 0 = DISABLE 1 = ENABLE
23	0x0	DPAUX_INT_C1MASK: 0 = DISABLE 1 = ENABLE
21	0x0	SOR_INT_C1MASK: 0 = DISABLE 1 = ENABLE
20	0x0	TSEC_INT_C1MASK: 0 = DISABLE 1 = ENABLE
19	0x0	MSENC_INT_C1MASK: 0 = DISABLE 1 = ENABLE
16	0x0	DSIB_INT_C1MASK: 0 = DISABLE 1 = ENABLE
13	0x0	VIC_INT_C1MASK: 0 = DISABLE 1 = ENABLE
12	0x0	DSI_INT_C1MASK: 0 = DISABLE 1 = ENABLE
10	0x0	HDMI_INT_C1MASK: 0 = DISABLE 1 = ENABLE
9	0x0	DISPLAYB_INT_C1MASK: 0 = DISABLE 1 = ENABLE
8	0x0	DISPLAY_INT_C1MASK: 0 = DISABLE 1 = ENABLE
2	0x0	VI_INT_C1MASK: 0 = DISABLE 1 = ENABLE
0	0x0	HOST_INT_C1MASK: 0 = DISABLE 1 = ENABLE

12.9.5 HOST1X_SYNC_HINTSTATUS_0

Host Interrupt Status

Contains the interrupt status for the various host interrupts. The status is sticky and is PENDING until cleared (write 1's to INTSTATUS to clear).

Offset: 0x20 │ Read/Write: R/W │ Reset: 0xX0000000 (0bx00xxxxxxxx00000x00000000000000)

Bit	R/W	Reset	Description
31	RO	X	HINTSTATUS_EXT_INT: Additional interrupts pending in the HINSTATUS_EXT register 0 = NOT_PENDING 1 = PENDING
30	RW	0x0	TIMER_INTP: Timer Interrupt from the Protected Channel

Bit	R/W	Reset	Description
			0 = NOT_PENDING 1 = PENDING
29	RW	0x0	CSW_HOST1XW2MC_INT: Host write client FIFO has filled up
19	RW	0x0	RDMA_DATABUF_THOLD_INT0: Read DMA data FIFO in port0 reached high level watermark 0 = NOT_PENDING 1 = PENDING
18	RW	0x0	RDMA_BUF_THOLD_INT0: Buffer threshold reached in read DMA port0 0 = NOT_PENDING 1 = PENDING
17	RW	0x0	RDMA_BUF_OFLOW_INT0: Buffer overflow in read DMA port0 0 = NOT_PENDING 1 = PENDING
16	RW	0x0	RDMA_INVAL_CLREQ_INT: Invalid client request to read DMA 0 = NOT_PENDING 1 = PENDING
15	RW	0x0	XBAR_TSEC_TIMEOUT_ID: ID of the CPU which timed out and updated the timeout address 0 = TSEC 1 = XBAR
13	RW	0x0	UNIT2_ACTMON_INTR: Unit2 ACTMON generates interrupt 0 = NOT_PENDING 1 = PENDING
12	RW	0x0	UNIT1_ACTMON_INTR: Unit1 ACTMON generates interrupt 0 = NOT_PENDING 1 = PENDING
11	RW	0x0	WAIT_INT11: WAIT has completed on channel 11 0 = NOT_PENDING 1 = PENDING
10	RW	0x0	WAIT_INT10: WAIT has completed on channel 10 0 = NOT_PENDING 1 = PENDING
9	RW	0x0	WAIT_INT9: WAIT has completed on channel 9 0 = NOT_PENDING 1 = PENDING
8	RW	0x0	WAIT_INT8: WAIT has completed on channel 8 0 = NOT_PENDING 1 = PENDING
7	RW	0x0	WAIT_INT7: WAIT has completed on channel 7 0 = NOT_PENDING 1 = PENDING
6	RW	0x0	WAIT_INT6: WAIT has completed on channel 6 0 = NOT_PENDING 1 = PENDING
5	RW	0x0	WAIT_INT5: WAIT has completed on channel 5 0 = NOT_PENDING 1 = PENDING
4	RW	0x0	WAIT_INT4: WAIT has completed on channel 4 0 = NOT_PENDING 1 = PENDING
3	RW	0x0	WAIT_INT3: WAIT has completed on channel 3 0 = NOT_PENDING 1 = PENDING
2	RW	0x0	WAIT_INT2: WAIT has completed on channel 2

Bit	R/W	Reset	Description
			0 = NOT_PENDING 1 = PENDING
1	RW	0x0	WAIT_INT1: WAIT has completed on channel 1 0 = NOT_PENDING 1 = PENDING
0	RW	0x0	WAIT_INT0: WAIT has completed on channel 0 0 = NOT_PENDING 1 = PENDING

12.9.6 HOST1X_SYNC_HINTMASK_0

Host Interrupt Mask

Contains the interrupt mask bits for all of the host interrupts. If the INT_MASK is ENABLED for a particular module, the modules INT bit from INTSTATUS contributes to the global interrupt signal.

Offset: 0x24 │ Read/Write: R/W │ Reset: 0x00000000 (0b000xxxxxxxxx0000xx00000000000000)

Bit	Reset	Description
31	0x0	HINTSTATUS_EXT_INTMASK: 0 = DISABLE 1 = ENABLE
30	0x0	TIMER_INTMASKP: Timer Interrupt Mask for the Protected Channel 0 = DISABLE 1 = ENABLE
29	0x0	CSW_HOST1XW2MC_INTMASK: 0 = DISABLE 1 = ENABLE
19	0x0	RDMA_DATABUF_THOLD_INTMASK0: 0 = DISABLE 1 = ENABLE
18	0x0	RDMA_BUF_THOLD_INTMASK0: 0 = DISABLE 1 = ENABLE
17	0x0	RDMA_BUF_OFLOW_INTMASK0: 0 = DISABLE 1 = ENABLE
16	0x0	RDMA_INVAL_CLREQ_INTMASK: 0 = DISABLE 1 = ENABLE
13	0x0	UNIT2_ACTMON_INTRMASK: 0 = DISABLE 1 = ENABLE
12	0x0	UNIT1_ACTMON_INTRMASK: 0 = DISABLE 1 = ENABLE
11	0x0	WAIT_INTMASK11: 0 = DISABLE 1 = ENABLE
10	0x0	WAIT_INTMASK10: 0 = DISABLE 1 = ENABLE
9	0x0	WAIT_INTMASK9: 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
8	0x0	WAIT_INTMASK8: 0 = DISABLE 1 = ENABLE
7	0x0	WAIT_INTMASK7: 0 = DISABLE 1 = ENABLE
6	0x0	WAIT_INTMASK6: 0 = DISABLE 1 = ENABLE
5	0x0	WAIT_INTMASK5: 0 = DISABLE 1 = ENABLE
4	0x0	WAIT_INTMASK4: 0 = DISABLE 1 = ENABLE
3	0x0	WAIT_INTMASK3: 0 = DISABLE 1 = ENABLE
2	0x0	WAIT_INTMASK2: 0 = DISABLE 1 = ENABLE
1	0x0	WAIT_INTMASK1: 0 = DISABLE 1 = ENABLE
0	0x0	WAIT_INTMASK0: 0 = DISABLE 1 = ENABLE

12.9.7 HOST1X_SYNC_HINTSTATUS_EXT_0

Extended Host Interrupt Status

Contains additional interrupt status bits that did not fit in the HINTSTATUS register. When any of these bits is set, the HINTSTATUS_EXT_INT bit will also be set in the HINSTATUS register. Each status bit is sticky and PENDING until cleared (write 1's to HINTSTATUS_EXT to clear).

Offset: 0x28 │ Read/Write: R/W │ Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
31	0x0	IP_WRITE_INT: Write transaction timeout occurred after IP_BUSY_TIMEOUT cycles. Offending address in IP_WRITE_TIMEOUT_ADDR. 0 = NOT_PENDING 1 = PENDING
30	0x0	IP_READ_INT: Read transaction timeout occurred after IP_BUSY_TIMEOUT cycles. Offending address in IP_READ_TIMEOUT_ADDR. 0 = NOT_PENDING 1 = PENDING
11	0x0	CMDPP_ILLEGAL_OPCODE_INT11: CMDPP11 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
10	0x0	CMDPP_ILLEGAL_OPCODE_INT10: CMDPP10 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING

Bit	Reset	Description
9	0x0	CMDPP_ILLEGAL_OPCODE_INT9: CMDPP9 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
8	0x0	CMDPP_ILLEGAL_OPCODE_INT8: CMDPP8 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
7	0x0	CMDPP_ILLEGAL_OPCODE_INT7: CMDPP7 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
6	0x0	CMDPP_ILLEGAL_OPCODE_INT6: CMDPP6 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
5	0x0	CMDPP_ILLEGAL_OPCODE_INT5: CMDPP5 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
4	0x0	CMDPP_ILLEGAL_OPCODE_INT4: CMDPP4 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
3	0x0	CMDPP_ILLEGAL_OPCODE_INT3: CMDPP3 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
2	0x0	CMDPP_ILLEGAL_OPCODE_INT2: CMDPP2 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
1	0x0	CMDPP_ILLEGAL_OPCODE_INT1: CMDPP1 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
0	0x0	CMDPP_ILLEGAL_OPCODE_INT0: CMDPP0 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING

12.9.8 HOST1X_SYNC_HINTMASK_EXT_0

Offset: 0x2c │ Read/Write: R/W │ Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
31	0x0	IP_WRITE_INTMASK: 0 = DISABLE 1 = ENABLE
30	0x0	IP_READ_INTMASK: 0 = DISABLE 1 = ENABLE
11	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK11: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
10	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK10: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
9	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK9: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
8	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK8: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
7	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK7: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
6	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK6: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
5	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK5: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
4	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK4: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
3	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK3: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
2	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK2: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
1	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK1: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
0	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK0: Mask CMDPP_ILLEGAL_OPCODE_INT0 interrupt bit. 0 = DISABLE 1 = ENABLE

12.9.9 HOST1X_SYNC_CF_SETUPDONE_0

Write to this register to trigger an update of the FIFO's pointers. **ONLY DO THIS WHEN THE FIFO IS EMPTY.**

Offset: 0xa0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	CF_SETUPDONE: Dummy bit

12.9.10 HOST1X_SYNC_CMDPROC_CTRL_0

Offset: 0xa4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0xx00x)

Bit	Reset	Description
5	0x0	INTFC_CLKEN_OVR
2	0x0	GATHER_PARSE_DISABLED
1	0x0	DROP_ILLEGAL_OPCODES

12.9.11 HOST1X_SYNC_CMDPROC_STAT_0

Offset: 0xa8 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:0	X	ILLEGAL_OPCODE

12.9.12 HOST1X_SYNC_CMDPROC_STOP_0

CH*_CMDPROC_STOP stops issuing commands from the command FIFO. This is useful to stop other channels when a channel teardown is needed to prevent unwanted traffic from happening at the same time.

Offset: 0xac │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11	0x0	CH11_CMDPROC_STOP: 0 = RUN 1 = STOP
10	0x0	CH10_CMDPROC_STOP: 0 = RUN 1 = STOP
9	0x0	CH9_CMDPROC_STOP: 0 = RUN 1 = STOP
8	0x0	CH8_CMDPROC_STOP: 0 = RUN 1 = STOP
7	0x0	CH7_CMDPROC_STOP: 0 = RUN 1 = STOP
6	0x0	CH6_CMDPROC_STOP: 0 = RUN 1 = STOP
5	0x0	CH5_CMDPROC_STOP: 0 = RUN 1 = STOP
4	0x0	CH4_CMDPROC_STOP: 0 = RUN 1 = STOP
3	0x0	CH3_CMDPROC_STOP: 0 = RUN 1 = STOP
2	0x0	CH2_CMDPROC_STOP: 0 = RUN 1 = STOP
1	0x0	CH1_CMDPROC_STOP: 0 = RUN 1 = STOP
0	0x0	CH0_CMDPROC_STOP: 0 = RUN 1 = STOP

12.9.13 HOST1X_SYNC_CH_TEARDOWN_0

Channel teardown register. Tells the hardware that a channel has gone away. Will reset that channel's command FIFO and release any locks it has in the arbiter. Will NOT reset that channel's output FIFO, which can be emptied by reading out all remaining entries.

Offset: 0xb0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11	X	CH11_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN

Bit	Reset	Description
10	X	CH10_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
9	X	CH9_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
8	X	CH8_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
7	X	CH7_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
6	X	CH6_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
5	X	CH5_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
4	X	CH4_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
3	X	CH3_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
2	X	CH2_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
1	X	CH1_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
0	X	CH0_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN

12.9.14 HOST1X_SYNC_MOD_TEARDOWN_0

Module teardown register. If a module is reset, the host needs to reset its state with respect to which channels own that module. Whenever a module is reset, the corresponding teardown bit in this register should be written. Module teardown will only work if the command FIFO and command processor are in a good state (the FIFO is empty of traffic for that channel and the command processor is at an opcode boundary). If an entire channel needs to be reset, the CH_TEARDOWN register should be used instead.

Offset: 0xb4 │ Read/Write: R/W │ Reset: 0x0XXXXX0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
26	X	ISPB_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
24	X	ISP_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
23	X	DPAUX_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN

Bit	Reset	Description
21	X	SOR_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
20	X	TSEC_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
19	X	MSENC_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
16	X	DSIB_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
13	X	VIC_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
12	X	DSI_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
10	X	HDMI_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
9	X	DISPLAYB_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
8	X	DISPLAY_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
2	X	VI_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN

12.9.15 HOST1X_SYNC_DISPLAY_STATUS_0

The per-client status registers indicate which channel owns each client as well as the current working class for each client. This is useful for determining each client's state with regard to granting context switches. _CURRCL is the current class ID for that module.

Offset: 0xdc │ Read/Write: RO │ Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	DISPLAY_CURRCL

12.9.16 HOST1X_SYNC_DISPLAYB_STATUS_0

Offset: 0xe0 │ Read/Write: RO │ Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	DISPLAYB_CURRCL

12.9.17 HOST1X_SYNC_ISP_STATUS_0

Offset: 0xe4 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	ISP_CURRCL

12.9.18 HOST1X_SYNC_DSI_STATUS_0

Offset: 0xe8 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	DSI_CURRCL

12.9.19 HOST1X_SYNC_HDMI_STATUS_0

Offset: 0xec | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	HDMI_CURRCL

12.9.20 HOST1X_SYNC_SOR_STATUS_0

Offset: 0xf0 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	SOR_CURRCL

12.9.21 HOST1X_SYNC_DPAUX_STATUS_0

Offset: 0xf4 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	DPAUX_CURRCL

12.9.22 HOST1X_SYNC_VI_STATUS_0

Offset: 0xf8 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	VI_CURRCL

12.9.23 HOST1X_SYNC_DSIB_STATUS_0

Offset: 0xfc | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	DSIB_CURRCL

12.9.24 HOST1X_SYNC_VIC_STATUS_0

Offset: 0x100 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description

Bit	Reset	Description
25:16	X	VIC_CURRCL

12.9.25 HOST1X_SYNC_MSENC_STATUS_0

Offset: 0x104 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	MSENC_CURRCL

12.9.26 HOST1X_SYNC_TSEC_STATUS_0

Offset: 0x108 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	TSEC_CURRCL

12.9.27 HOST1X_SYNC_ISPB_STATUS_0

Offset: 0x10c | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	ISPB_CURRCL

12.9.28 HOST1X_SYNC_DIRECT_MODULE_CONFIG_0

Offset: 0x1a0 | Read/Write: R/W | Reset: 0x54000000 (0b01010100000000000000000000000xx)

Bit	Reset	Description
31:2	0x15000000	BASE

12.9.29 HOST1X_SYNC_USEC_CLK_0

Number of host clocks needed to make a microsecond. Used for the DELAY host method. For example, if the host clock is 250 MHz, this register should be programmed to 250. If the host clock is 150 MHz, this register should be programmed to 150.

Offset: 0x1a4 | Read/Write: R/W | Reset: 0x0000015e (0bxxxxxxxxxxxxxxxxxxxx000101011110)

Bit	Reset	Description
11:0	0x15e	USEC_CLKS

12.9.30 HOST1X_SYNC_CTXSW_TIMEOUT_CFG_0

When a channel writes to a new class on a client, it generates a context switch. To keep from continually switching contexts if channels are addressing the same client, this register is used, so that the Host1x does not switch until the channel has either received WAIT_CTXSW_CNT clocks before switching or the channel stops targeting the particular client. If two channels are accessing two classes of a client, to guarantee a channel sending multiple commands for a class, this register can be used.

Offset: 0x1a8 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxx00001111)

Bit	Reset	Description
7:0	0xf	WAIT_CTXSW_CNT: Number of cycles to wait

12.9.31 HOST1X_SYNC_INDREG_DMA_CTRL_0

Enable use of DMA engine to move data from indirect register interface to memory. ATTN_LVL controls flow between host and DMA. DMA will not start a transfer until the set number of slots are full:

- 00 = 1 slot
- 01 = 4 slots
- 10 = 8 slots

Offset: 0x1ac │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx000x0000)

Bit	Reset	Description
7	0x0	AHBDMA_ENABLE: Enable generation of request to DMA engine 0 = DISABLE 1 = ENABLE
6:5	0x0	AHBDMA_ATTN_LVL: Number of entries to receive before sending DMA request, 1, 4, or 8
3:0	0x0	AHBDMA_CHID: channel being used by indirect read for DMA (which chout FIFO to monitor)

12.9.32 HOST1X_SYNC_CHANNEL_PRIORITY_0

Set channel priority for dual ring arbitration (hi/lo). Used in arbitrating MLOCKs.

Offset: 0x1b0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11	0x0	HIPRI_CH11: 0 = DISABLE 1 = ENABLE
10	0x0	HIPRI_CH10: 0 = DISABLE 1 = ENABLE
9	0x0	HIPRI_CH9: 0 = DISABLE 1 = ENABLE
8	0x0	HIPRI_CH8: 0 = DISABLE 1 = ENABLE
7	0x0	HIPRI_CH7: 0 = DISABLE 1 = ENABLE
6	0x0	HIPRI_CH6: 0 = DISABLE 1 = ENABLE
5	0x0	HIPRI_CH5: 0 = DISABLE 1 = ENABLE
4	0x0	HIPRI_CH4: 0 = DISABLE 1 = ENABLE
3	0x0	HIPRI_CH3: 0 = DISABLE 1 = ENABLE
2	0x0	HIPRI_CH2: 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
1	0x0	HIPRI_CH1: 0 = DISABLE 1 = ENABLE
0	0x0	HIPRI_CH0: 0 = DISABLE 1 = ENABLE

12.9.33 HOST1X_SYNC_CDMA_ASM_TIMEOUT_0

Timeout value determines how long the assembly logic will wait before it flushes data from the data FIFO to avoid head-of-line blocking.

Note: Do not use a value of zero -- causes immediate flushing so no forward progress is made.

Offset: 0x1b4 │ Read/Write: R/W │ Reset: 0x00000404 (0bxxxxxxxxxxxxxxxxxxxx00100xxx00100)

Bit	Reset	Description
12:8	0x4	CDMA_ASM_GFIFO_TIMEOUT
4:0	0x4	CDMA_ASM_DFIFO_TIMEOUT

12.9.34 HOST1X_SYNC_CDMA_MISC_0

CDMA_DELAY_PUT

Delay put_addr updates. This can potentially increase MC performance slightly by gathering multiple consecutive put_addr updates into 1 update, which reduces the amount of requests to the MC. In addition, it can be used to cover a race condition where a write to memory has not completed before CDMA starts fetching the data. But in this case, the counter has to be set to a high value, which *will* affect performance.

CDMA_SIMPLE_PREFETCH

Reduce per-channel requests to 1 every 3 cycles. This can reduce performance in 2 ways:

- The memory request FIFO will fill up slightly slower, but the impact will be minimal.
- It will cause the channel arbiter to switch to other channels when they have a request available, which will reduce locality. On the plus side, it has much less nasty corner cases.

CDMA_PUT_SYNC_DISABLE

CDMA_EN_STATS

Enable statistics counters.

- - Count the number of 128-bit words fetched by CDMA
- - Count the number of 128-bit words thrown away by CDMA

This allows us to get an idea how efficient the CDMA really is when there are multiple push-buffers active at the same time.

Offset: 0x1b8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxx0x000xx000000)

Bit	Reset	Description
12	0x0	CDMA_EN_STATS
10	0x0	CDMA_CLKEN_OVR

Bit	Reset	Description
9	0x0	CDMA_PUT_SYNC_DISABLE
8	0x0	CDMA_SIMPLE_PREFETCH
5:0	0x0	CDMA_DELAY_PUT

12.9.35 HOST1X_SYNC_IP_BUSY_TIMEOUT_0

Offset: 0x1bc │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	IP_BUSY_TIMEOUT: Number of busy cycles before requesting a retry. 0 = disabled

12.9.36 HOST1X_SYNC_IP_READ_TIMEOUT_ADDR_0

Offset: 0x1c0 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:2	X	IP_READ_TIMEOUT_ADDR: Address of transaction that caused an AXI read timeout

12.9.37 HOST1X_SYNC_IP_WRITE_TIMEOUT_ADDR_0

Offset: 0x1c4 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:2	X	IP_WRITE_TIMEOUT_ADDR: Address of transaction that caused an AXI write timeout

12.9.38 HOST1X_SYNC_MCCIF_THCTRL_0

Memory write client FIFO status. Reads out the available 128-bit entries. If writing through the buffered frame buffer write region, writes are accumulated up to 128 bits before being flushed, so 10 entries can mean up to 40 writes. Memory client high-priority threshold control register. Sets the threshold of when the client becomes high priority.

Offset: 0x1d8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx000000xxxxxxxx)

Bit	R/W	Reset	Description
13:8	RW	0x0	CSW_HOST1XW2MC_HPTH
5:0	RO	X	CSW_HOST1XW_FIFOSTAT

12.9.39 HOST1X_SYNC_HC_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control (where applicable) and Clock Gating Control Register

Note: This FIFO timing aspects of this register are no longer supported, but are retained for software compatibility.

The clock override/ovr_mode fields of this register control the 2nd-level clock gating for the client and MC side of the MCCIF. All clock gating is enabled by default.

A '1' written to the rclk/wclk override field results in one of the following:

- With wclk/rclk override mode = LEGACY, the clock reverts to legacy mode of operation where the clock is on whenever the client clock is enabled.

- With wclk/rclk override mode = ON, the clock is always on inside the MCCIF and PC.

A '1' written to the cclk override field keeps the client's clock always on inside the MCCIF.

Offset: 0x1dc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxx00000xxxxxxxxxxxx0000)

Bit	Reset	Description
20	LEGACY	HC_RCLK_OVR_MODE: 0 = LEGACY 1 = ON
19	LEGACY	HC_WCLK_OVR_MODE: 0 = LEGACY 1 = ON
18	0x0	HC_CCLK_OVERRIDE
17	0x0	HC_RCLK_OVERRIDE
16	0x0	HC_WCLK_OVERRIDE
3	DISABLE	HC_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	HC_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	HC_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	HC_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

12.9.40 HOST1X_SYNC_TIMEOUT_WCOAL_HC_0

Write Coalescing Time-Out Register

Note: Write coalescing is no longer supported by the MCCIF clients. Registers are retained for software compatibility but are not used by the hardware.

Offset: 0x1e0 │ Read/Write: R/W │ Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00110010)

Bit	Reset	Description
7:0	0x32	HOST1XW_WCOAL_TMVAL

12.9.41 HOST1X_SYNC_MLOCK_0_0

MLOCK Registers

MLOCK is 1 whenever someone holds the lock, and is 0 otherwise. MLOCK is normally set and cleared using the host methods ACQUIRE_MLOCK and RELEASE_MLOCK.

If a CPU wants to acquire a lock, then it reads this register. If the value returned is 0, it has successfully acquired the MLOCK. A return value of 1 indicates that the acquire failed. At any time, the CPU can write 0 to MLOCK, releasing the MLOCK (note that hardware does not check if the CPU owned the lock). If the CPU writes a 1 to MLOCK, this is undefined and is ignored by the hardware (a NOP). Use MLOCK_OWNER to read the status of a lock (since reading MLOCK itself has a side effect).

Offset: 0x2c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_0

12.9.42 HOST1X_SYNC_MLOCK_1_0

Offset: 0x2c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_1

12.9.43 HOST1X_SYNC_MLOCK_2_0

Offset: 0x2c8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_2

12.9.44 HOST1X_SYNC_MLOCK_3_0

Offset: 0x2cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_3

12.9.45 HOST1X_SYNC_MLOCK_4_0

Offset: 0x2d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_4

12.9.46 HOST1X_SYNC_MLOCK_5_0

Offset: 0x2d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_5

12.9.47 HOST1X_SYNC_MLOCK_6_0

Offset: 0x2d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_6

12.9.48 HOST1X_SYNC_MLOCK_7_0

Offset: 0x2dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_7

12.9.49 HOST1X_SYNC_MLOCK_8_0

Offset: 0x2e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_8

12.9.50 HOST1X_SYNC_MLOCK_9_0

Offset: 0x2e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_9

12.9.51 HOST1X_SYNC_MLOCK_10_0

Offset: 0x2e8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_10

12.9.52 HOST1X_SYNC_MLOCK_11_0

Offset: 0x2ec | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_11

12.9.53 HOST1X_SYNC_MLOCK_12_0

Offset: 0x2f0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_12

12.9.54 HOST1X_SYNC_MLOCK_13_0

Offset: 0x2f4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_13

12.9.55 HOST1X_SYNC_MLOCK_14_0

Offset: 0x2f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_14

12.9.56 HOST1X_SYNC_MLOCK_15_0

Offset: 0x2fc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_15

12.9.57 HOST1X_SYNC_MLOCK_OWNER_0_0

MLOCK_OWNER is a read-only status for MLOCK. When MLOCK_*_OWNS are all zeros, it indicates that the MLOCK is free (zero). When MLOCK has been acquired (set), then one of MLOCK_*_OWNS will be non-zero.

Either bit 0 or 1 will be set -- indicating whether a channel or a CPU has acquired the MLOCK. If a channel owns the MLOCK, then the channel number is given by the MLOCK_OWNER_CHID field.

Offset: 0x340 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_0
1	X	MLOCK_CPU_OWNS_0
0	X	MLOCK_CH_OWNS_0

12.9.58 HOST1X_SYNC_MLOCK_OWNER_1_0

Offset: 0x344 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_1
1	X	MLOCK_CPU_OWNS_1
0	X	MLOCK_CH_OWNS_1

12.9.59 HOST1X_SYNC_MLOCK_OWNER_2_0

Offset: 0x348 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_2
1	X	MLOCK_CPU_OWNS_2
0	X	MLOCK_CH_OWNS_2

12.9.60 HOST1X_SYNC_MLOCK_OWNER_3_0

Offset: 0x34c │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_3
1	X	MLOCK_CPU_OWNS_3
0	X	MLOCK_CH_OWNS_3

12.9.61 HOST1X_SYNC_MLOCK_OWNER_4_0

Offset: 0x350 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_4
1	X	MLOCK_CPU_OWNS_4
0	X	MLOCK_CH_OWNS_4

12.9.62 HOST1X_SYNC_MLOCK_OWNER_5_0

Offset: 0x354 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_5
1	X	MLOCK_CPU_OWNS_5
0	X	MLOCK_CH_OWNS_5

12.9.63 HOST1X_SYNC_MLOCK_OWNER_6_0

Offset: 0x358 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_6
1	X	MLOCK_CPU_OWNS_6
0	X	MLOCK_CH_OWNS_6

12.9.64 HOST1X_SYNC_MLOCK_OWNER_7_0

Offset: 0x35c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_7
1	X	MLOCK_CPU_OWNS_7
0	X	MLOCK_CH_OWNS_7

12.9.65 HOST1X_SYNC_MLOCK_OWNER_8_0

Offset: 0x360 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_8
1	X	MLOCK_CPU_OWNS_8
0	X	MLOCK_CH_OWNS_8

12.9.66 HOST1X_SYNC_MLOCK_OWNER_9_0

Offset: 0x364 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_9
1	X	MLOCK_CPU_OWNS_9
0	X	MLOCK_CH_OWNS_9

12.9.67 HOST1X_SYNC_MLOCK_OWNER_10_0

Offset: 0x368 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_10
1	X	MLOCK_CPU_OWNS_10
0	X	MLOCK_CH_OWNS_10

12.9.68 HOST1X_SYNC_MLOCK_OWNER_11_0

Offset: 0x36c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_11
1	X	MLOCK_CPU_OWNS_11
0	X	MLOCK_CH_OWNS_11

12.9.69 HOST1X_SYNC_MLOCK_OWNER_12_0

Offset: 0x370 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_12
1	X	MLOCK_CPU_OWNS_12
0	X	MLOCK_CH_OWNS_12

12.9.70 HOST1X_SYNC_MLOCK_OWNER_13_0

Offset: 0x374 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_13
1	X	MLOCK_CPU_OWNS_13
0	X	MLOCK_CH_OWNS_13

12.9.71 HOST1X_SYNC_MLOCK_OWNER_14_0

Offset: 0x378 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_14
1	X	MLOCK_CPU_OWNS_14
0	X	MLOCK_CH_OWNS_14

12.9.72 HOST1X_SYNC_MLOCK_OWNER_15_0

Offset: 0x37c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_15
1	X	MLOCK_CPU_OWNS_15
0	X	MLOCK_CH_OWNS_15

12.9.73 HOST1X_SYNC_MLOCK_ERROR_0_0

If a channel attempts to release an MLOCK that it does not own, then the release has no effect on MLOCK, but the corresponding error bit is set (this includes releasing an MLOCK owned by no one).

Offset: 0x3c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15	0x0	MLOCK_ERROR_15
14	0x0	MLOCK_ERROR_14
13	0x0	MLOCK_ERROR_13
12	0x0	MLOCK_ERROR_12
11	0x0	MLOCK_ERROR_11
10	0x0	MLOCK_ERROR_10
9	0x0	MLOCK_ERROR_9
8	0x0	MLOCK_ERROR_8
7	0x0	MLOCK_ERROR_7
6	0x0	MLOCK_ERROR_6
5	0x0	MLOCK_ERROR_5
4	0x0	MLOCK_ERROR_4
3	0x0	MLOCK_ERROR_3
2	0x0	MLOCK_ERROR_2
1	0x0	MLOCK_ERROR_1
0	0x0	MLOCK_ERROR_0

12.9.74 HOST1X_SYNC_SYNCPT_BASE_n_0

Syncpt base registers are used by wait_syncpt_base method. The wait will be released when SYNCPT[indx] >= (BASE[base_indx] + offset), where indx, base_indx, and offset are supplied by the wait method.

There are 64 Syncpt base registers, where n = 0 through 63.

Offset: 0x600 + (n * 0x4) │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BASE_n

12.10 Host Class Methods

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

Class offsets are always relative to the class (based at 0).

12.10.1 NV_CLASS_HOST_INCR_SYNCPT_0

All Classes have the INCR_SYNCPT method. For host, this method, immediately increments SYNCPT[indx], irrespective of the condition. Note that INCR_SYNCPT_CNTRL and INCR_SYNCPT_ERROR are included for consistency with host clients, but writes to INCR_SYNCPT_CNTRL have no effect on the operation of Host1x, and because there are no condition FIFOs to overflow, INCR_SYNCPT_ERROR will never be set.

Offset: 0x0 │ Byte Offset: 0x0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = COND_4 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15 16 = COND_16 17 = COND_17 18 = COND_18 19 = COND_19 20 = COND_20 21 = COND_21 22 = COND_22 23 = COND_23 24 = COND_24 25 = COND_25 26 = COND_26 27 = COND_27 28 = COND_28 29 = COND_29 30 = COND_30 31 = COND_31
7:0	0x0	INDX: syncpt index value

12.10.2 NV_CLASS_HOST_INCR_SYNCPT_CNTRL_0

Offset: 0x1 | Byte Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal states of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all Host1x clients affected, then clear all SOFT_RESETs.

12.10.3 NV_CLASS_HOST_INCR_SYNCPT_ERROR_0

Offset: 0x2 | Byte Offset: 0x8 | Read/Write: R/W | Reset: 0xXXXXXXXX (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.

12.10.4 NV_CLASS_HOST_WAIT_SYNCPT_0

Wait on syncpt method.

Command dispatch will stall until:

SYNCPT[indx][NV_HOST1X_SYNCPT_THRESH_WIDTH-1:0] >= threshold[NV_HOST1X_SYNCPT_THRESH_WIDTH-1:0]

The comparison takes into account the possibility of wrapping. Note that more bits are allocated for index and threshold than may be used in an implementation. Use NV_HOST1X_SYNCPT_NB_PTS for the number of syncpts, and NV_HOST1X_SYNCPT_THRESH_WIDTH for the number of bits used by the comparison.

Offset: 0x8 | Byte Offset: 0x20 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	INDX
23:0	X	THRESH

12.10.5 NV_CLASS_HOST_WAIT_SYNCPT_BASE_0

Wait on syncpt method using base register.

Command dispatch will stall until:

SYNCPT[indx][NV_HOST1X_SYNCPT_THRESH_WIDTH-1:0] >= (SYNCPT_BASE[base_indx]+offset)

The comparison takes into account the possibility of wrapping. Note that more bits are allocated for INDX and BASE_INDX than may be used in an implementation. Use NV_HOST1X_SYNCPT_NB_PTS for the number of syncpts, Use NV_HOST1X_SYNCPT_NB_BASES for the number of syncpt_bases, and NV_HOST1X_SYNCPT_THRESH_WIDTH for the number of bits used by the comparison If NV_HOST1X_SYNCPT_THRESH_WIDTH is greater than 16, the offset is sign-extended before it is added to SYNCPT_BASE.

Offset: 0x9 | Byte Offset: 0x24 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	INDX

Bit	Reset	Description
23:16	X	BASE_INDX
15:0	X	OFFSET

12.10.6 NV_CLASS_HOST_WAIT_SYNCPT_INCR_0

Wait on syncpt increment method.

Command dispatch will stall until the next time that SYNCPT[indx] is incremented.

Note that more bits are allocated for INDX than may be used in an implementation. Use NV_HOST1X_SYNCPT_NB_PTS for the number of syncpts.

Offset: 0xa │ Byte Offset: 0x28 │ Read/Write: R/W │ Reset: 0xXX000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	INDX

12.10.7 NV_CLASS_HOST_LOAD_SYNCPT_BASE_0

Load syncpt base method.

SYNCPT_BASE[indx] = value

Offset: 0xb │ Byte Offset: 0x2c │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	BASE_INDX
23:0	X	VALUE

12.10.8 NV_CLASS_HOST_INCR_SYNCPT_BASE_0

Increment syncpt base method.

SYNCPT_BASE[indx] += offset

Offset: 0xc │ Byte Offset: 0x30 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	BASE_INDX
23:0	X	OFFSET

12.10.9 NV_CLASS_HOST_CLEAR_0

Clear method. Any bits set in VECTOR will be cleared in the channel's RAISE vector.

Offset: 0xd │ Byte Offset: 0x34 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VECTOR

12.10.10 NV_CLASS_HOST_WAIT_0

Wait method. Command dispatch will stall until any of the bits set in VECTOR become set in the channel's RAISE vector.

Offset: 0xe │ Byte Offset: 0x38 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VECTOR

12.10.11 NV_CLASS_HOST_WAIT_WITH_INTR_0

Wait with Interrupt method. Identical to the WAIT method except an interrupt will be triggered when the WAIT requirement is satisfied. This is obsolete and preserved here only for completeness.

Offset: 0xf │ Byte Offset: 0x3c │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VECTOR

12.10.12 NV_CLASS_HOST_DELAY_USEC_0

Delay number of microseconds. Command dispatch will stall until the number of microseconds indicated in NUSEC has passed. The timing of microseconds is controlled by the USEC_CLK register.

Offset: 0x10 │ Byte Offset: 0x40 │ Read/Write: R/W │ Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:0	X	NUSEC: Enough for 1.05 seconds

12.10.13 NV_CLASS_HOST_TICKCOUNT_HI_0

This register value will initialize the high 32 bits of the tick count value in the host clock counter.

Offset: 0x11 │ Byte Offset: 0x44 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_HI: Read or write tick count

12.10.14 NV_CLASS_HOST_TICKCOUNT_LO_0

This register value will initialize the low 32 bits of the tick count value in the host clock counter.

Offset: 0x12 │ Byte Offset: 0x48 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_LO: Read or write tick count

12.10.15 NV_CLASS_HOST_TICKCTRL_0

This register write enables the tick counter on the host clock to start counting.

Offset: 0x13 │ Byte Offset: 0x4c │ Read/Write: R/W │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	TICKCNT_ENABLE: Enable or Disable tick counter 0 = DISABLE 1 = ENABLE

12.10.16 NV_CLASS_HOST_INDCTRL_0

Indirect Addressing

These registers (along with INDDATA) are used to indirectly read/write either register or memory. Host registers are not accessible using this interface. If AUTOINC is set, INDOFFSET increments by 4 on every access of INDDATA.

Either INDCTRL/INDOFF2 or INDOFF can be used, but INDOFF may not be able to address all memory in chips with large memory maps. The redundant bits in INDCTRL and INDOFF are shared, so writing either offset sets those bits.

Note: The following restrictions apply to the use of indirect memory writes:

- At initialization time, do a dummy indirect write (with all byte enables set to zero)
- Dedicate an MLOCK for indirect memory writes, then before a channel issues a set of indirect memory writes it must acquire this MLOCK; after the writes have been issued, the MLOCK is released -- this will restrict the use of indirect memory writes to a single channel at a time.

Offset: 0x2b │ Byte Offset: 0xac │ Read/Write: R/W │ Reset: 0xXX00000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	INDBE: Byte enables. Will apply to all subsequent data transactions. Not applicable for reads.
27	X	AUTOINC: Auto increment of read/write address 0 = DISABLE 1 = ENABLE
26	X	SPOOL: Route return data to spool FIFO, only applicable to reads 0 = DISABLE 1 = ENABLE
1	X	ACCTYPE: Access type: indirect register or indirect frame buffer 0 = REG 1 = FB
0	X	RWN: Read/write 0 = WRITE 1 = READ

12.10.17 NV_CLASS_HOST_INDOFF2_0

Offset: 0x2c │ Byte Offset: 0xb0 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:18	X	INDMODID: ACCTYPE=REG: Register module ID 0 = HOST1X 2 = VI 8 = DISPLAY 9 = DISPLAYB 10 = HDMI 11 = TVO 12 = DSI 13 = VIC 16 = DSIB 19 = MSENC 20 = TSEC 21 = SOR 23 = DPAUX 24 = ISP 26 = ISPB
31:2	X	INDOFFSET: ACCTYPE=FB: frame buffer address
17:2	X	INDROFFSET: ACCTYPE=REG: register offset ([15:0])

12.10.18 NV_CLASS_HOST_INDOFF_0

Offset: 0x2d │ Byte Offset: 0xb4 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	INDBE: Byte enables. Will apply to all subsequent data transactions. Not applicable for reads.
27	X	AUTOINC: Auto increment of read/write address 0 = DISABLE 1 = ENABLE
26	X	SPOOL: Route return data to spool FIFO, only applicable to reads 0 = DISABLE 1 = ENABLE
25:18	X	INDMODID: ACCTYPE=REG: register module ID 0 = HOST1X 2 = VI 8 = DISPLAY 9 = DISPLAYB 10 = HDMI 11 = TVO 12 = DSI 13 = VIC 16 = DSIB 19 = MSENC 20 = TSEC 21 = SOR 23 = DPAUX 24 = ISP 26 = ISPB
25:2	X	INDOFFSET: ACCTYPE=FB: frame buffer address
17:2	X	INDROFFSET: ACCTYPE=REG: register offset ([15:0])
1	X	ACCTYPE: Access type: indirect register or indirect frame buffer 0 = REG 1 = FB
0	X	RWN: Read/write 0 = WRITE 1 = READ

12.10.19 NV_CLASS_HOST_INDDATA_0

These registers, when written, either write to the data to the INDOFFSET in INDOFF or trigger a read of the offset at INDOFFSET. This is an array of 31 identical register entries; the register fields below apply to each entry.

Offset: 0x2e..0x4c │ Byte Offset: 0xb8..0x130 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INDDATA: Read or write data

12.10.20 NV_CLASS_HOST_LOAD_SYNCPT_PAYLOAD_32_0

Offset: 0x4e │ Byte Offset: 0x138 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CHANNEL_SYNCPT_PAYLOAD

12.10.21 NV_CLASS_HOST_STALLCTRL_0

Offset: 0x4f | Byte Offset: 0x13c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0X0	ENABLE

12.10.22 NV_CLASS_HOST_WAIT_SYNCPT_32_0

Offset: 0x50 | Byte Offset: 0x140 | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	INDX

12.10.23 NV_CLASS_HOST_WAIT_SYNCPT_BASE_32_0

Offset: 0x51 | Byte Offset: 0x144 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:8	X	BASE_INDX
7:0	X	INDX

12.10.24 NV_CLASS_HOST_LOAD_SYNCPT_BASE_32_0

Offset: 0x52 | Byte Offset: 0x148 | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	INDX

12.10.25 NV_CLASS_HOST_INCR_SYNCPT_BASE_32_0

Offset: 0x53 | Byte Offset: 0x14c | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	INDX

12.10.26 NV_CLASS_HOST_STALLCOUNT_HI_0

Offset: 0x54 | Byte Offset: 0x150 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0X0	STALLCOUNT_HI

12.10.27 NV_CLASS_HOST_STALLCOUNT_LO_0

Offset: 0x55 | Byte Offset: 0x154 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0X0	STALLCOUNT_LO

12.10.28 NV_CLASS_HOST_XFERCTRL_0

Offset: 0x56 | Byte Offset: 0x158 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0X0	ENABLE

12.10.29 NV_CLASS_HOST_CHANNEL_XFER_HI_0

Offset: 0x57 | Byte Offset: 0x15c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0X0	CHANNEL_XFER_HI

12.10.30 NV_CLASS_HOST_CHANNEL_XFER_LO_0

Offset: 0x58 | Byte Offset: 0x160 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0X0	CHANNEL_XFER_LO

12.11 Host Proto Channel Registers

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

12.11.1 HOST1X_PROTCHANNEL_FIFOSTAT_0

CFNUMEMPTY is the number of free slots available in the per-channel command FIFO (needed for PIO or polling for completion of a wait).

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

Bit	Reset	Description
31	X	INDRDY: Indicates that INDCOUNT==0, so it should be OK to issue another read
28:24	X	OUTFENTRIES: Number of entries available for reading in this channel's output FIFO
20:16	X	REGFNUMEMPTY: Register write/read FIFO free count
12	X	CFGATHER: Indicates whether or not GATHER is active. If a GATHER command issued via PIO, software must wait for the GATHER to be IDLE before issuing another command. 0 = IDLE 1 = BUSY
11	X	CFEMPTY: Indicates whether the command FIFO is empty or not 0 = NOTEMPTY 1 = EMPTY
10:0	X	CFNUMEMPTY: Command FIFO free count

12.11.2 HOST1X_PROTCHANNEL_INDOFF_0

The INDOFF and INDOFF2 registers (along with INDCNT and INDDATA) are used to indirectly read/write modules outside the host. If AUTOINC is set, INDOFFSET increments by 4 on every access of INDDATA. REGFNUMEMPTY is polled to determine when valid data can be read from INDDATA.

The INDOFF register has limited capability on chips with large memory maps. If the top bit of the memory address is >= 27, all of memory cannot be addressed with INDOFF. In these cases, use INDOFF2 to set the offset while still using INDOFF to set the other parameters. Always have INDOFFUPD set to NO_UPDATE in these cases. For register accesses, using INDOFF (with INDOFFUPD set to UPDATE) is always more efficient, since it only requires one write.

Indirect frame buffer writes are STRONGLY DISCOURAGED. There are better ways to write to memory (direct and through the channel memory map) and there is limited flow control in the host. It is very easy to get into trouble with indirect frame buffer writes.

Offset: 0x4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	AUTOINC: Auto increment of read/write address 0 = DISABLE 1 = ENABLE
30	X	ACCTYPE: access type: indirect register or indirect frame buffer 0 = REG 1 = FB
29	X	BUF32B: Buffer up 32 bits of register data before sending it. Otherwise, register writes will be sent as soon as they are received. Does not support byte writes in 16-bit host. Does not affect frame buffer writes. 0 = NOBUF 1 = BUF
28:27	X	INDSWAP: Indirect frame buffer access swap control. 00 = No byte swap 01 = 16-bit byte swap ([31:0] -> {[23:16],[31:24],[7:0],[15:8]}) 10 = 32-bit byte swap ([31:0] -> {[7:0],[15:8],[23:16],[31:24]}) 11 = 32-bit word swap ([31:0] -> {[15:8],[7:0],[31:24],[23:16]}) 0 = NONE 1 = BYTE16 2 = BYTE32 3 = WORD32
25:18	X	INDMODID: ACCTYPE=REG: register module ID 0 = HOST1X 2 = VI 8 = DISPLAY 9 = DISPLAYB 10 = HDMI 12 = DSI 13 = VIC 16 = DSIB 19 = MSENC 20 = TSEC 21 = SOR 23 = DPAUX 24 = ISP 26 = ISPB
25:2	X	INDOFFSET: ACCTYPE=FB: frame buffer address
17:2	X	INDROFFSET: ACCTYPE=REG: register offset ([15:0])
0	X	INDOFFUPD: Optionally disable the update of INDOFFSET when writing this register 0 = UPDATE 1 = NO_UPDATE

12.11.3 HOST1X_PROTCHANNEL_INDCNT_0

Indirect register access count

Used to trigger indirect reads. Holds the number of registers/memory locations that will be read out. Channels should not request more than there is space available in their output FIFO. Only the protected channel should make liberal use of this feature for speeding up context switching.

For indirect frame buffer reads, each channel cannot issue more than NV_HOST1X_MAX_IND_FB_READS at once. The read data must return and be written into the per-channel output FIFO before any additional reads can be issued.

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0X0	INDCOUNT

12.11.4 HOST1X_PROTCHANNEL_INDDATA_0

This register, when written, writes the data to the INDOFFSET in INDOFF. For reads, a REGFNUMEMPTY number of 32-bit values can be read before needing to poll FIFOSTAT again.

The per-channel output FIFO (OUTFENTRIES) is readable via this offset. A read of INDDATA will pop an entry off of the per-channel output FIFO.

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

Bit	Reset	Description
31:0	0X0	INDDATA: Read or write data

12.11.5 HOST1X_PROTCHANNEL_INDOFF2_0

Note: This spec file contains additions to the "common" registers that cannot be put in the common section, otherwise all of the other registers will shift.

The INDOFF and INDOFF2 registers (along with INDCNT and INDDATA) are used to indirectly read/write modules outside the host. If AUTOINC is set, INDOFFSET increments by 4 on every access of INDDATA. REGFNUMEMPTY is polled to determine when valid data can be read from INDDATA.

The INDOFF register has limited capability on chips with large memory maps. If the top bit of the memory address is >= 27, all of memory cannot be addressed with INDOFF. In these cases, use INDOFF2 to set the offset while still using INDOFF to set the other parameters. Always have INDOFFUPD set to NO_UPDATE in these cases. For register accesses, using INDOFF (with INDOFFUPD set to UPDATE) is always more efficient, since it only requires one write.

Indirect frame buffer writes are STRONGLY DISCOURAGED. There are better ways to write to memory (direct and through the channel memory map) and there is limited flow control in the host. It is very easy to get into trouble with indirect frame buffer writes.

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

Bit	Reset	Description
25:18	X	INDMODID2: ACCTYPE=REG: register module ID 0 = HOST1X 2 = VI 8 = DISPLAY 9 = DISPLAYB 10 = HDMI 12 = DSI 13 = VIC 16 = DSIB 19 = MSENC 20 = TSEC 21 = SOR 23 = DPAUX 24 = ISP 26 = ISPB
31:2	X	INDOFFSET2: ACCTYPE=FB: frame buffer address
17:2	X	INDROFFSET2: ACCTYPE=REG: register offset ([15:0])

12.11.6 HOST1X_PROTCHANNEL_TICKCOUNT_HI_0

This register holds the high 32 bits of the tick count value

Offset: 0x90 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_HI

12.11.7 HOST1X_PROTCHANNEL_TICKCOUNT_LO_0

This register holds the low 32 bits of the tick count value

Offset: 0x94 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_LO

12.11.8 HOST1X_PROTCHANNEL_CHANNELCTRL_0

This register will be used for controlling some channel-related commands including enabling/disabling of the tick counter and enabling/disabling of Kernel command filtering from the gather buffers.

Offset: 0x98 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0x0)

Bit	Reset	Description
2	DISABLE	KERNEL_FILTER_GBUFFER: Enable setclass command filter for gather buffers 0 = DISABLE 1 = ENABLE
0	DISABLE	ENABLETICKCNT: Enable or disable tick counter 0 = DISABLE 1 = ENABLE

12.11.9 HOST1X_PROTCHANNEL_PAYLOAD_0

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

Bit	Reset	Description
31:0	0x0	PAYLOAD

12.11.10 HOST1X_PROTCHANNEL_STALLCTRL_0

Offset: 0xa0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ENABLE_CHANNEL_STALL

12.11.11 HOST1X_PROTCHANNEL_STALLCOUNT_HI_0

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

Bit	Reset	Description
31:0	0x0	STALLCOUNT_HI

12.11.12 HOST1X_PROTCHANNEL_STALLCOUNT_LO_0

Offset: 0xa8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	STALLCOUNT_LO

12.11.13 HOST1X_PROTCHANNEL_XFERCTRL_0

Offset: 0xac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ENABLE_CHANNEL_XFER

12.11.14 HOST1X_PROTCHANNEL_CHANNEL_XFER_HI_0

Offset: 0xb0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	CHANNEL_XFER_HI

12.11.15 HOST1X_PROTCHANNEL_CHANNEL_XFER_LO_0

Offset: 0xb4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	CHANNEL_XFER_LO

12.11.16 HOST1X_PROTCHANNEL_HOST1X_CHANNEL_SPARE_0

Offset: 0xb8 | Read/Write: R/W | Reset: 0x00ff0000 (0b00000000111111111000000000000000)

Bit	Reset	Description
31:16	0xff	CHANNEL_SPARE_HI
15:0	0x0	CHANNEL_SPARE_LO

12.11.17 HOST1X_PROTCHANNEL_TICKCOUNT_THRESHOLD_HI_0

This register holds the high 32 bits of the tick count threshold value

Offset: 0xbc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_THRESHOLD_HI

12.11.18 HOST1X_PROTCHANNEL_TICKCOUNT_THRESHOLD_LO_0

This register holds the low 32 bits of the tick count threshold value

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_THRESHOLD_LO

12.11.19 HOST1X_PROTCHANNEL_TICKCOUNT_THRESHOLD_CTRL_0

Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	DISABLE	THRES_CMP: Enable or disable tick threshold compare 0 = DISABLE 1 = ENABLE

13.0 VIDEO IMAGE COMPOSITOR (VIC)

The VIC is the Video Image Compositor unit. It is also referred to as the Video Scalar (VS) unit. The VIC unit is suitable for use in mobile systems. It implements video post-processing functions needed by a video playback application to produce the final image for the player window.

The compositor implements much of the DirectX Video Acceleration 2.0 Enhanced Video Processor specification, including deinterlacing, scaling, color conversion, proc-amp, and compositing for up to 5 input surfaces. It supports advanced features like histogram correction, gamma correction, and non-linear color enhancement.

13.1 Features

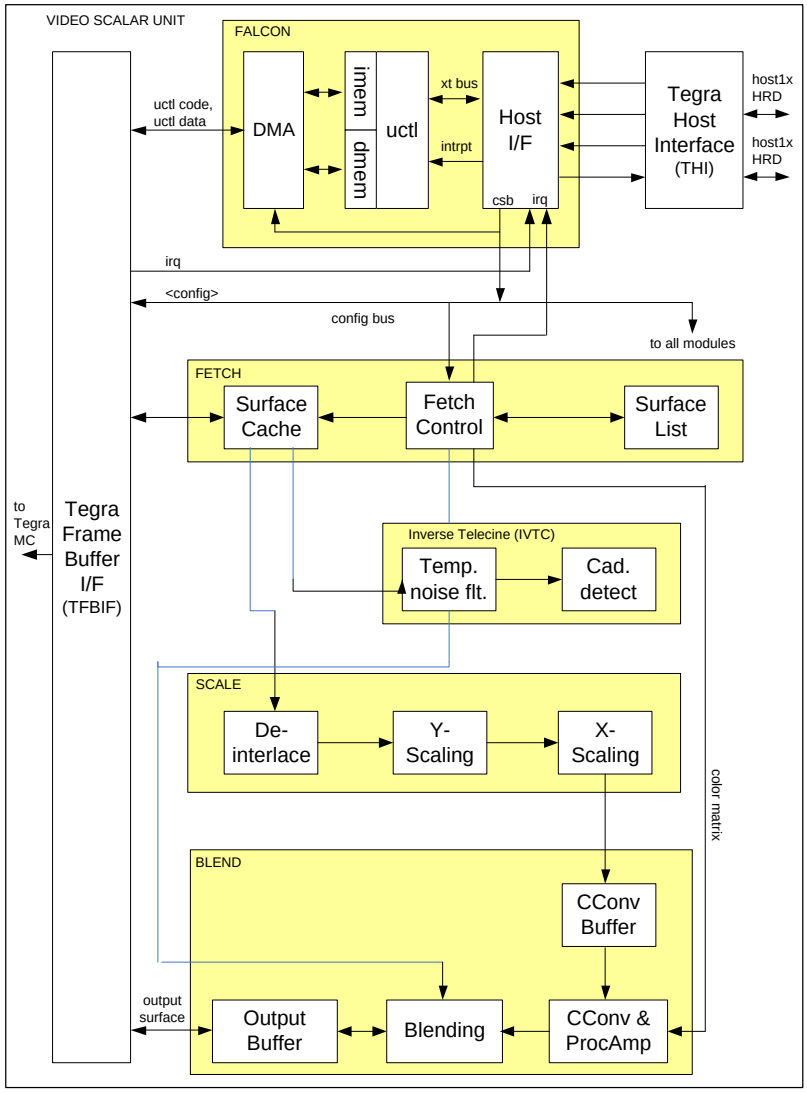
The VIC implements the following features:

- Deinterlacing
- Inverse Teleciné
- Scale
- Color Conversion
- Memory Format Conversion
- Blend/Composite
- Stereo Pixel Interleave
- Rotation

13.2 Block Diagram Description

The following figure shows the top-level block diagram of the Video Image Compositor.

Figure 22: VIC Block Diagram



13.2.1 Falcon Microprocessor

The NVIDIA Falcon Microcontroller is used for method processing and performing processing related to inverse telecine.

13.2.2 Tegra Host Interface (THI)

The Tegra Host Interface (THI) shim handles incoming methods from the host1x bus and passes those methods to the Falcon microcontroller, in addition to handling Tegra sync points.

13.2.3 Tegra Frame Buffer Interface (TFBIF)

The Tegra Frame Buffer Interface (TFBIF) interfaces with the Tegra K1 memory system. The TFBIF has one read and one write MCCIF client.

13.2.4 Fetch Unit

During preprocessing (IVTC passes) the fetch control breaks the output surfaces into 16x16 byte tiles and processes columns of these tiles at a time.

During main processing, the fetch control breaks the output surface up into 64x16 pixel tiles and processes those individually. For each tile, the fetch unit loops over all input surfaces.

When processing each tile, 64*16+*n* pixels need to be fetched, where *n* is the extra padding needed for scaling and deinterlacing. Each tile being fetched is stored in the Surface Cache. This cache holds 128 entries of 256 bytes.

13.2.5 Inverse Telecine (IVTC) Unit

The IVTC unit runs on individual video streams at a time and consists of 2 separate passes.

The first pass does noise reduction and motion buffer updates. It also calculates the weave and artifact counts used for the IVTC decision. The final decision for IVTC is done in Falcon microcode.

The second pass combines the updated motion buffer and the previous motion buffer (which has a different parity) and combines them for DiSi1 deinterlacing during the main processing pass.

13.2.6 Scale Unit

The Scale Unit consists of three blocks:

- The first block deinterlaces the input. The deinterlacing modes supported are DiSi1, BOB, and WEAVE. Weave is only used on progressive content or when inverse telecine detected a cadence.
- The second block scales the data in the vertical direction producing at most 16 lines of output to the next stage.
- The third block scales the data in the horizontal direction producing at most 64 pixels of output.

13.2.7 Blend Unit

The Blend Unit consists of four blocks:

- The first block buffers the incoming data and converts from any input order into interleaved pixel order (either ARGB or AYUV).
- The second block performs all color conversion tasks.
- The third block blends data into the output buffer. This stage is also used during preprocessing.
- The fourth block is an input-output DMA that can read original buffer data from the memory system and write back the final blended result. The reading is only used during preprocessing.

13.3 Functionality

13.3.1 Slots

The VIC supports up to five different input pictures that can be composited on to a single surface.

Support for five input pictures enables composition of multiple video streams from Blu-ray video content in a single pass:

- Background
- Primary Video
- Secondary Video
- Presentation Graphics
- Interactive Graphics

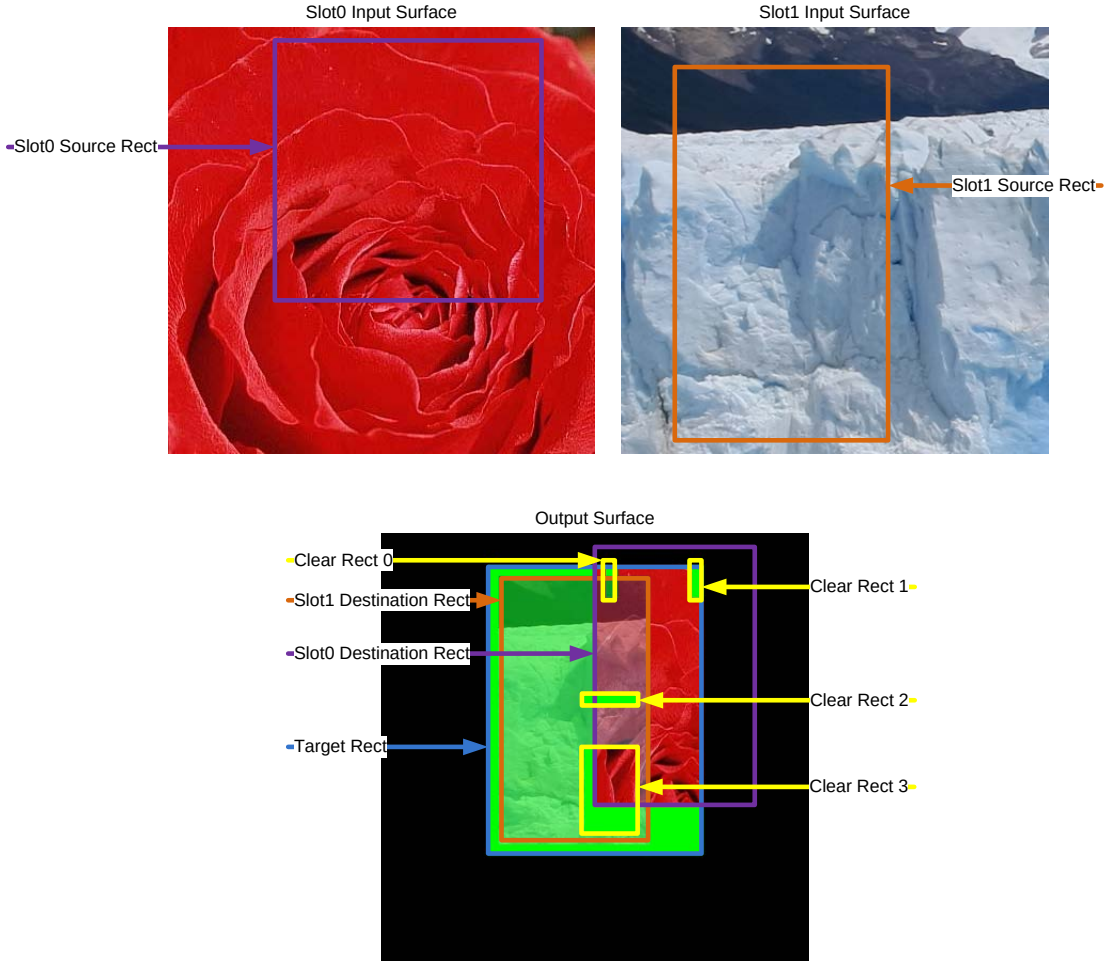
Each slot has a state that is specified through the config structure. Control parameters, such as the input formats, deinterlacing, scaling factors, clipping, color-space conversion, are independently set per input surface. All slots support the same functionality.

During the composition stage, the slots are blended in ascending order (that is, the output of slot1 will be blended onto slot 0, the output of slot 2 is blended on to slot 1, etc.).

Each slot can be programmed to access multiple surfaces. The number of surfaces used depends on the type of slot input (e.g., interlaced versus progressive), input format (e.g., packed versus planar color formats), and also on the type of processing (e.g., temporal noise reduction versus cadence detection versus deinterlacing).

13.3.1.1 Surface Composition

Figure 23: Programming Input and Output Surface Parameters



The maximum dimension of any image on the input or output sides of the VIC is 16384 pixels. The upper two bits of any 14-bit surface dimension will just be ignored.

Via the SurfaceListSurfaceStruct config structures, the dimensions of each input slot surface can be defined along with the pixel and memory format types. Note that the pixel format type must also be specified in the SurfaceCache0Struct config structure. A source rectangle defines the region of pixels of the input slot surface that will contribute to the composition of the output surface, and a destination rectangle defines the region of the output surface that is affected by a given input slot. Together, the source and destination rectangle parameters specify the scaling ratios desired. Each slot can also dictate how pixel data is laid out in each surface cache entry by specifying the amount of pixel data, i.e., the number of bytes wide, logically represented by each cache entry; this provides flexibility in reducing the memory overfetch associated with fetch of surfaces under various use cases (e.g., scaling ratios, deinterlacing modes).

Via the Blending0Struct config structure, the dimensions of the output surface can be defined along with the pixel and memory format types. Via the Blending0Struct and the SurfaceList0Struct config structs, a target rectangle can also be defined to restrict the pixel processing output to a certain rectangle in the output surface. A programmable color can also be set to fill the background of the target rectangle.

The VIC allows the programming of clear rectangles that prevent the fetch of pixels from specific rectangular regions of input slots. The use of the clear rectangles can be used to reduce redundant pixel fetches, for example, if an opaque surface is known to occlude a layer below it, a clear rectangle can be specified to prevent fetching of pixels from the occluded region of the lower layer. Up to eight clear rectangles are specified via the SurfaceListClearRectStruct config structs, and per-slot clear rect mask enables are specified via the SurfaceIList0Struct config struct, Clear rectangle coordinates are specified relative to the output surface base.

With the addition of the sub-pixel source rectangle feature, source rectangle coordinates are specified in the config structure in a U14.16 format, allowing the source rectangle to be specified at a sub-pixel resolution. Destination, target, and clear rectangles coordinates are pixel aligned.

13.3.2 Memory Format Support

The following memory formats must be supported on input and output. Input and output memory formats do not have to match.

- Pitch Linear
- Block Linear (16Bx2 kind with sector ordering)
  - 64x8 byte GOB format with 16x2 byte sectors
  - Contiguous 64B atoms are laid out as 32x2 byte blocks

The VIC Surface Cache supports the above memory formats on input when fetching pixel data from memory.

The VIC Output Buffer supports the above memory formats on input when reading background image data when pre-processing inverse telecine data from memory, and on the output side when writing composited image data back to memory.

Memory formats supported by the VIC are enumerated as follows:

Table 35 : Memory Format Enumerations

Surface Kind Enumeration	Encoding (Four Bits)
BLK_KIND_PITCH	0x0
BLK_KIND_GENERIC_16Bx2_TEGRA	0x1

To fully support the Block Linear format, block height should be parameterizable. The following set of enumerations is valid.

Table 36 : Block Height Enumerations

Block Height Enumeration	Value	Encoding (Four Bits, log2 Encoding)
ONE_GOB	1	0
TWO_GOBS	2	1
FOUR_GOBS	4	2
EIGHT_GOBS	8	3
SIXTEEN_GOBS	16	4

Block Height Enumeration	Value	Encoding (Four Bits, log2 Encoding)
THIRTYTWO_GOBS	32	5

13.3.2.1 Surface Transformation

The Fetch Control unit will be responsible for generating the source image addressing that will implement mirroring (across X-axis and/or Y-axis) of tiles within the Target rectangle of the output surface.

The output buffer can mirror pixels in the X and Y directions within each output tile. This ability in addition to the fetch control changes mentioned above gives the ability to flip the contents of the entire Target rectangle about its axis.

The output buffer also supports transposition of the Target Rectangle. This ability when combined with the target rectangle flips described above gives the ability to rotate images by any multiple of 90 degrees.

Rotation must be done in the pipeline after deinterlacing because the deinterlacer assumes a normal orientation of the input fields (interlaced field lines run horizontal). Rotation of the image prior to deinterlacing will make the deinterlacer ineffective.

Figure 24: Interlaced Field Lines Run Horizontally

The programmed starting address of the source and destination rectangles will remain at the upper left corner of the unit. The ‘Source Image’ diagrammed below shows the logical tile traversal pattern as the input rectangle traverses the VIC unit (tiles are read in raster order from left to right, then top to bottom), and the destination rectangle diagrammed below shows the expected transformations when the programmer specifies the combination of flipX, flipY, and transposeXY transformations available to be applied to the output rectangle.

Figure 25: Surface Transformation Orientations

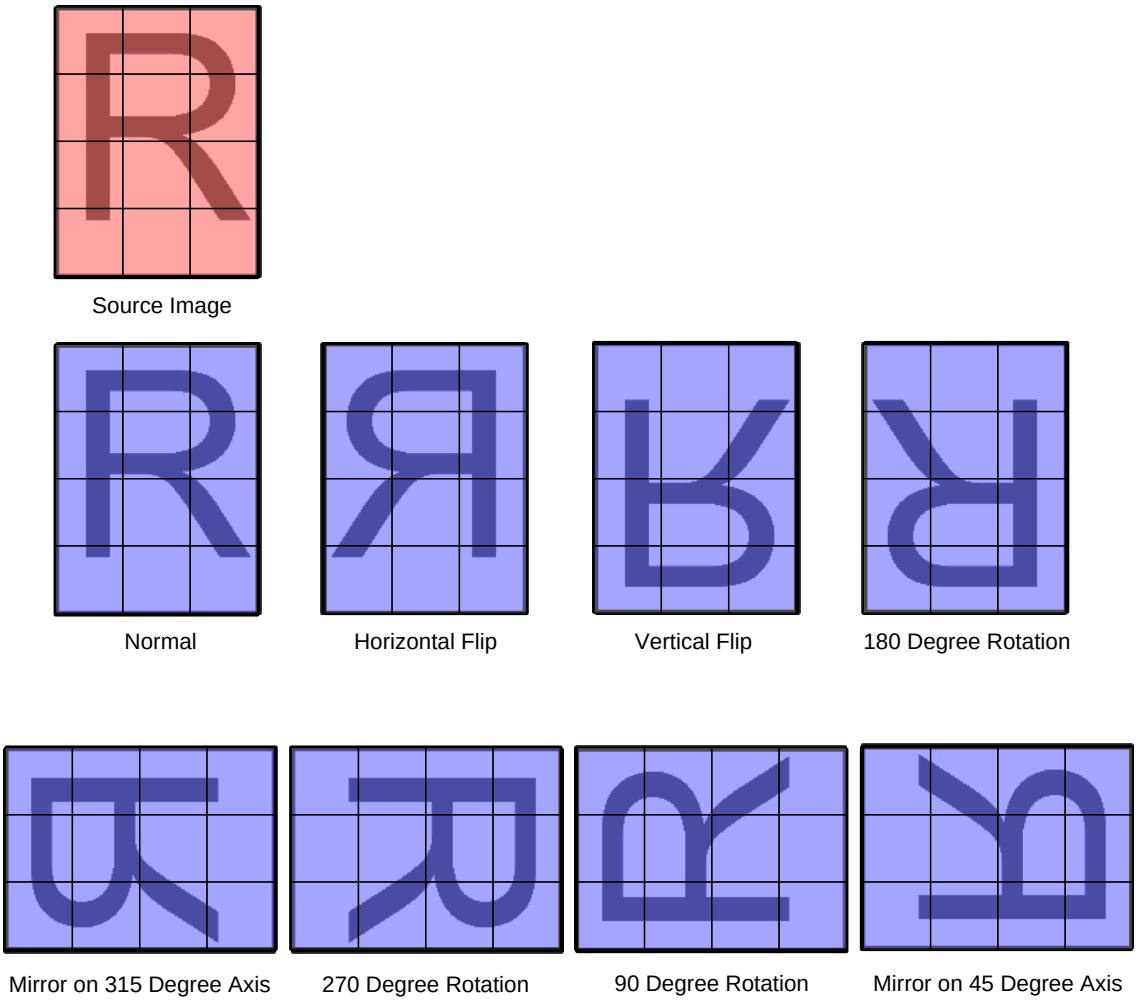


Table 37: Output Buffer Surface Transformation Controls

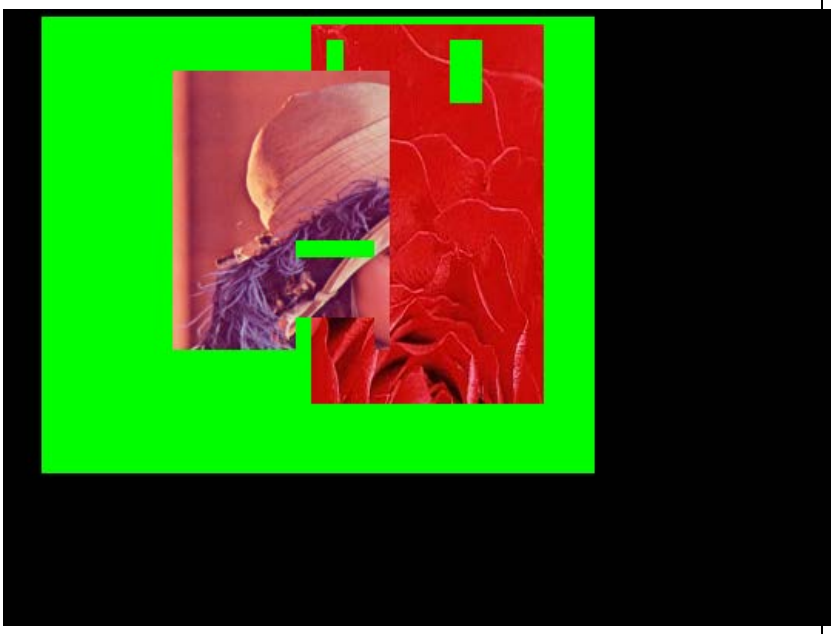
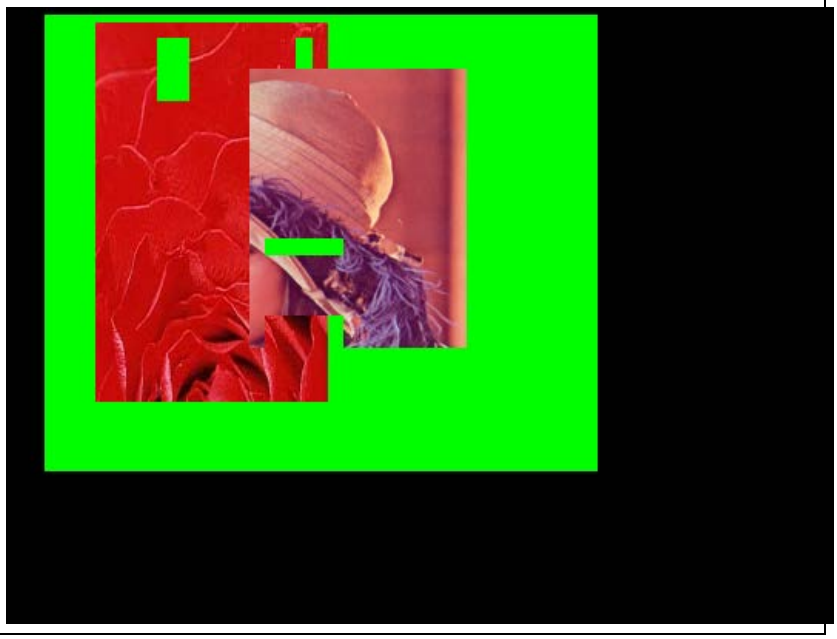
Transformation	transposeXY	flipY	flipX
Normal (no transformation)	0	0	0
H flip (mirror on vertical axis)	0	0	1
V flip (mirror on horizontal axis)	0	1	0
180-degree rotation	0	1	1
Mirror on 315-degree axis	1	0	0
270-degree rotation	1	1	0
90-degree rotation	1	0	1
Mirror on 45-degree axis	1	1	1

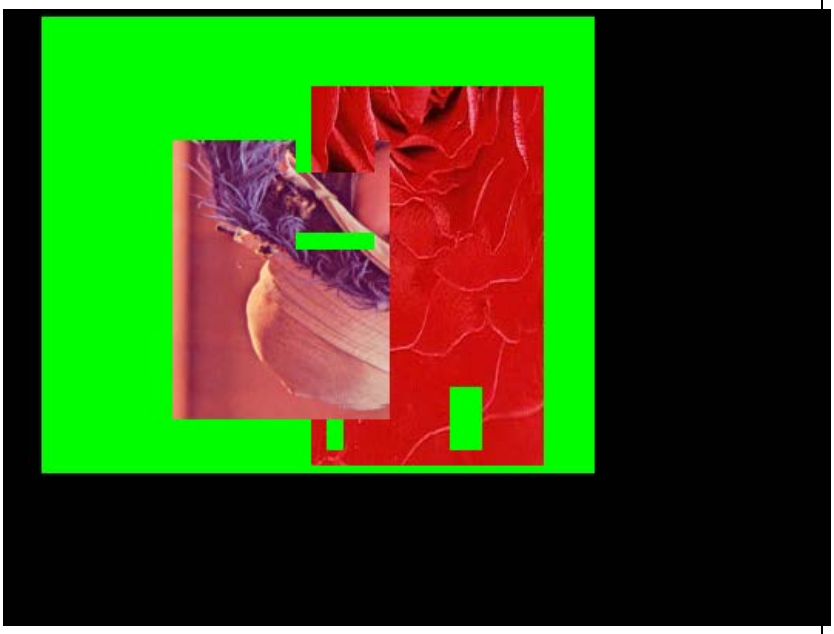
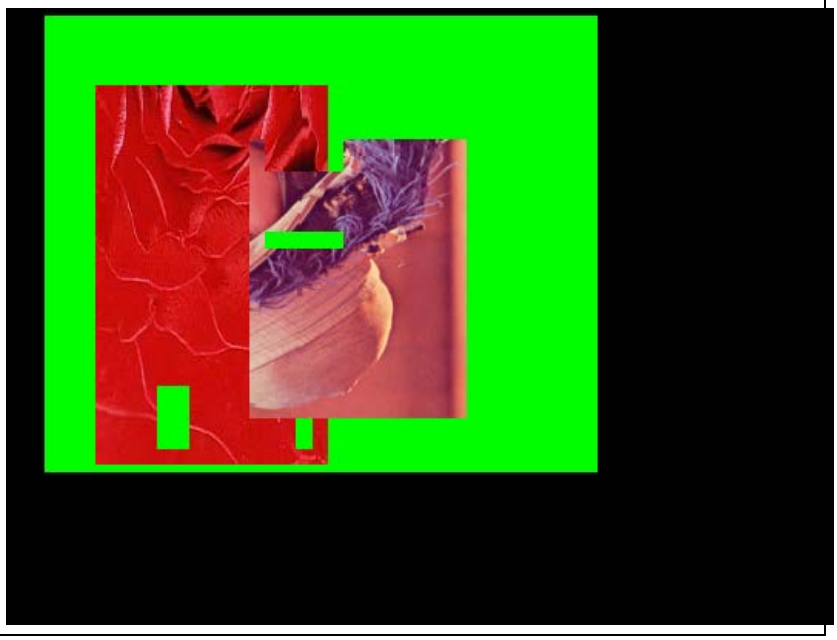
Surface mirroring (FlipX/FlipY) is accomplished via programming of the SurfaceList0Struct, FetchControl0Struct, and the Blending0Struct config structures. Surface transposition is accomplished via programming of the OutputTranspose field of the Blending0Struct config structure.

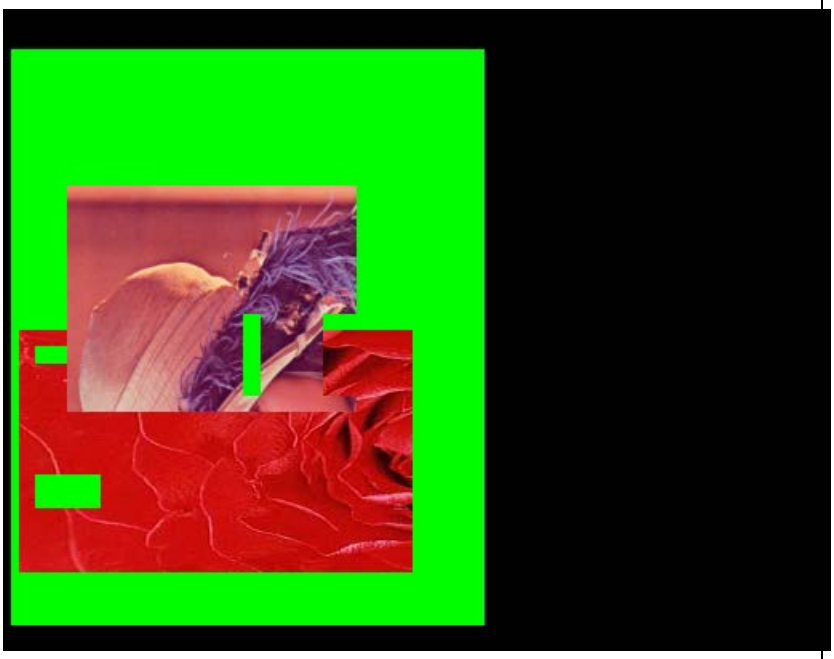
The table below shows the output image for a certain surface composition and how the output transformations affect the final output. The results of the transformations are summarized in the rules below.

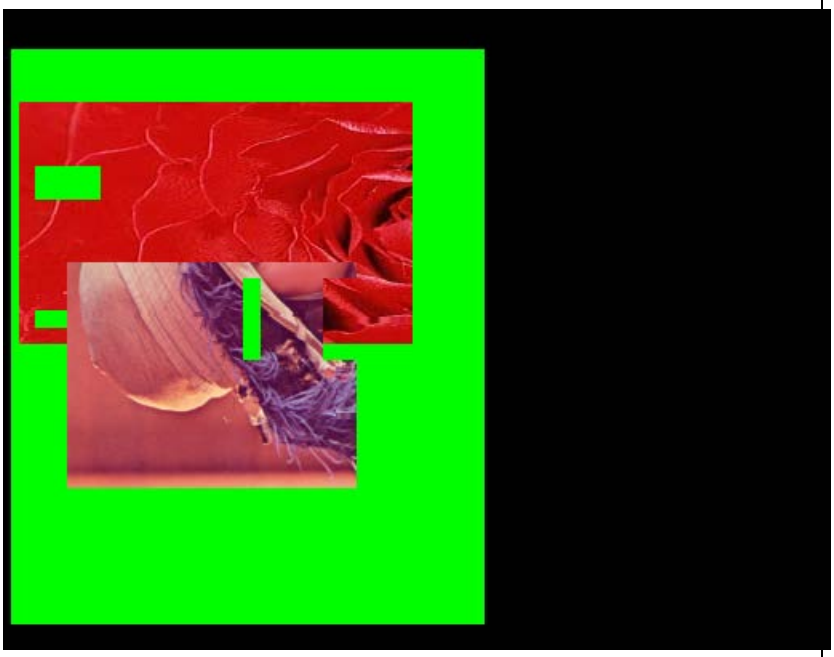
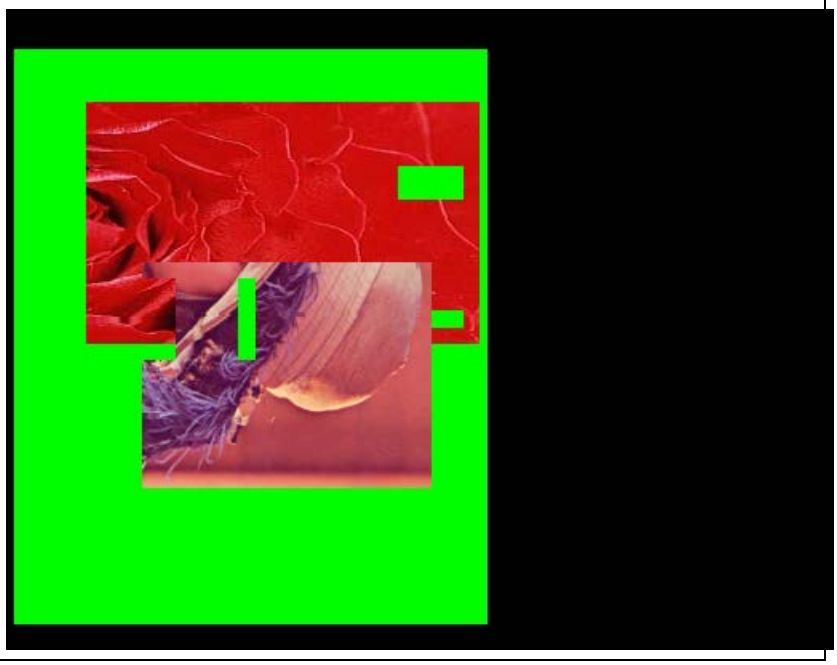
1. The output surface parameters (SurfaceWidth/Height, LumaWidth/Height, ChromaWidth/Height) are not affected by any of the transformations. It is software's responsibility to allocate a large enough buffer so that the target rectangle fits within the output surface even after transformation.
2. The target rectangle position in the output surface does NOT change when output flips (in X or Y) are enabled. However, the position of the target rectangle is transposed when OutputTranspose is enabled.
3. When output flips are enabled, the entire contents of the target rectangle are flipped, including all ClearRects and DestinationRects and their contents. The axis of the flip is the axis of the TargetRectangle.
4. When OutputTranspose is enabled, the position of the target rectangle is transposed, as well as all of its contents (including all clear/destination rectangles).
5. When transpose and flips are enabled together, the overall effect of the transformation happens as if the flip operations occur first, followed by the transpose operation.
6. When flip operations are enabled, every Destination Rectangle has to lie fully within the TargetRectangle. This rule has to be enforced to get consistent results while flipping because we define the flip operation as happening about the target rectangle axis, and clip after we do the flip.

Table 38: Output Buffer Surface Transformation Examples

Transformation	TransposeXY	FlipY	FlipX	Image
Normal (no transformation)	0	0	0	
H flip (mirror on vertical axis)	0	0	1	

Transformation	TransposeXY	FlipY	FlipX	Image
V flip (mirror on horizontal axis)	0	1	0	
180-degree rotation	0	1	1	

Transformation	TransposeXY	FlipY	FlipX	Image
Mirror on 315-degree axis	1	0	0	
270-degree rotation	1	1	0	

Transformation	TransposeXY	FlipY	FlipX	Image
90-degree rotation	1	0	1	
Mirror on 45-degree axis	1	1	1	

13.3.3 Memory Copy

The VIC memory copy bandwidth scales with the VIC core frequency. The VIC core pixel processing pipeline can output up to 4x2 components (2 pixels) per VIC clock.

13.3.4 Temporal Noise Reduction

VIC 1.0 implemented a temporal noise reduction algorithm based on a motion adaptive IIR filter, where the filter weight was adapted based on a few constants and the SOS difference between a 3x3 neighborhood of pixels around the current pixel in the current and previous frames.

The VIC 1.0 TNR algorithm (hereafter called TNR1) works well for videos or captures that have good lighting conditions and lower levels of noise in the input video. However, when the lighting conditions are poor and there is a lot of noise in the input video/capture, the algorithm tries to be safe by not modifying it much, and allows most of the noise to pass through into the output.

The TNR2 algorithm performs significantly better in the above mentioned cases by pre-filtering the data spatially and allowing significantly higher alpha values in the IIR blend. This algorithm can be turned on by programming the AdvancedDenoise* fields in the SurfaceCache0Struct. The TNR2 algorithm also requires information about the lighting conditions at which the video or capture is being shot (information that will probably be derived from ISP statistics), and this information should be programmed into the LightLevel fields in the SurfaceListSurfaceStructs (4 bits, 0-darkest, 15-brightest).

13.3.5 Inverse Telecine

There are two statistics that the VIC hardware calculates: the difference count and the artifact count. The VIC hardware just calculates the statistics; a state machine on the Falcon Microcontroller has to interpret these.

13.3.6 Deinterlacing

The VIC supports various modes of deinterlacing the input video content such as DiSi1, BOB, and WEAVE. Weave is only used on progressive content or in case inverse telecine detected a cadence.

13.3.7 Color Fill

Programming the output surface with only a background color and writing to memory will produce a color fill.

13.3.8 Bayer Pixel Formats

Bayer pixel data can come from camera sensor equipment in various sizes, for example, 6 bits, 10 bits, 14 bits, 16 bits. Most Bayer data is stored using existing pixel formats, such as T_L8, or T_L16. In some special cases, special formats are created to pack pixel data efficiently to work with existing memory formats, for example, T_X2Lc10Lb10La10, T_X2Le6Ld6Lc6Lb6La6.

13.3.9 Pixel Format Support

The VIC pipeline supports an increased number of pixel formats compared to what is currently comprehended by VIC1.0 on input and output. The VIC pipeline can accept a surface of a given input pixel format and produce a given output pixel format.

The table below lists the supported pixel formats. The symbols in color conversion mode column describe the internal VIC representation of the format, (e.g., 1P_1C represents 1 plane with 1 component pixels, 2P_1+2C represents 2 planes, 1 plane with 1 component pixels and a second plane with 2 component pixels).

Table 39: Pixel Format Support

Pixel Format	FOURCC	Input Support	Output Support	Rotation	Scaling	High Quality Deinterlacing	New index	Color_Conversion_Mode
T_A8		Y	Y	Y	Y	N	0	1P_1C
T_L8/ T_Y8/ T_YUV100		Y	Y	Y	Y	N	1	1P_1C
T_A4L4		Y	Y	Y	Y	N	2	1P_2C
T_L4A4		Y	Y	Y	Y	N	3	1P_2C
T_R8		Y	Y	Y	Y	N	4	1P_1C
T_A8L8		Y	Y	Y	Y	N	5	1P_2C
T_L8A8		Y	Y	Y	Y	N	6	1P_2C
T_R8G8		Y	Y	Y	Y	N	7	1P_2C
T_G8R8		Y	Y	Y	Y	N	8	1P_2C
T_B5G6R5		Y	Y	Y	Y	N	9	1P_3C
T_R5G6B5		Y	Y	Y	Y	N	10	1P_3C
T_B6G5R5		Y	Y	Y	Y	N	11	1P_3C
T_R5G5B6		Y	Y	Y	Y	N	12	1P_3C
T_A1B5G5R5		Y	Y	Y	Y	N	13	1P_4C
T_A1R5G5B5		Y	Y	Y	Y	N	14	1P_4C
T_B5G5R5A1		Y	Y	Y	Y	N	15	1P_4C
T_R5G5B5A1		Y	Y	Y	Y	N	16	1P_4C
T_A5B5G5R1		Y	Y	Y	Y	N	17	1P_4C
T_A5R1G5B5		Y	Y	Y	Y	N	18	1P_4C
T_B5G5R1A5		Y	Y	Y	Y	N	19	1P_4C
T_R1G5B5A5		Y	Y	Y	Y	N	20	1P_4C
T_X1B5G5R5		Y	Y	Y	Y	N	21	1P_4C
T_X1R5G5B5		Y	Y	Y	Y	N	22	1P_4C
T_B5G5R5X1		Y	Y	Y	Y	N	23	1P_4C
T_R5G5B5X1		Y	Y	Y	Y	N	24	1P_4C
T_A4B4G4R4		Y	Y	Y	Y	N	25	1P_4C
T_A4R4G4B4		Y	Y	Y	Y	N	26	1P_4C
T_B4G4R4A4		Y	Y	Y	Y	N	27	1P_4C
T_R4G4B4A4		Y	Y	Y	Y	N	28	1P_4C
T_A8B8G8R8		Y	Y	Y	Y	N	29	1P_4C
T_A8R8G8B8		Y	Y	Y	Y	N	30	1P_4C
T_B8G8R8A8		Y	Y	Y	Y	N	31	1P_4C
T_R8G8B8A8		Y	Y	Y	Y	N	32	1P_4C
T_X8B8G8R8		Y	Y	Y	Y	N	33	1P_4C
T_X8R8G8B8		Y	Y	Y	Y	N	34	1P_4C
T_B8G8R8X8		Y	Y	Y	Y	N	35	1P_4C

Pixel Format	FOURCC	Input Support	Output Support	Rotation	Scaling	High Quality Deinterlacing	New index	Color_Conversion_Mode
T_R8G8B8X8		Y	Y	Y	Y	N	36	1P_4C
T_A2B10G10R10		Y	N	Y	Y	N	37	1P_4C
T_A2R10G10B10		Y	N	Y	Y	N	38	1P_4C
T_B10G10R10A2		Y	N	Y	Y	N	39	1P_4C
T_R10G10B10A2		Y	N	Y	Y	N	40	1P_4C
T_A4P4		Y	N	Y	Y	N	41	1P_2C
T_P4A4		Y	N	Y	Y	N	42	1P_2C
T_P8A8		Y	N	Y	Y	N	43	1P_2C
T_A8P8		Y	N	Y	Y	N	44	1P_2C
T_P8		Y	N	Y	Y	N	45	1P_1C
T_P1		Y	N	Y	Y	N	46	1P_1C
T_U8V8		Y	Y	Y	Y	N	47	1P_2C
T_V8U8		Y	Y	Y	Y	N	48	1P_2C
T_A8Y8U8V8 (AYUV444 packed)		Y	Y	Y	Y	N	49	1P_4C
T_V8U8Y8A8 (AYUV444 packed)		Y	Y	Y	Y	N	50	1P_4C
T_Y8_U8__Y8_V8 (YUV422 packed)	YUY2/YUYV	Y	Y	Y	Y	N	51	1P_4C
T_Y8_V8__Y8_U8 (YUV422 packed)	YVYU	Y	Y	Y	Y	N	52	1P_4C
T_U8_Y8__V8_Y8 (YUV422 packed)	UYVY	Y	Y	Y	Y	N	53	1P_4C
T_V8_Y8__U8_Y8 (YUV422 packed)		Y	Y	Y	Y	N	54	1P_4C
T_Y8___U8V8_N444 (YUV444 semi-planar)		Y	Y	Y	Y	N	55	2P_1+2C
T_Y8___V8U8_N444 (YUV444 semi-planar)		Y	Y	Y	Y	N	56	2P_1+2C
T_Y8___U8V8_N422 (YUV422 semi-planar)	e.g., NV61	Y	Y	Y	Y	N	57	2P_1+2C
T_Y8___V8U8_N422 (YUV422 semi-planar)	e.g., NV16	Y	Y	Y	Y	N	58	2P_1+2C
T_Y8___U8V8_N422R (YUV422R semi-planar)		Y	Y	Y	Y	N	59	2P_1+2C
T_Y8___V8U8_N422R (YUV422R semi-planar)		Y	Y	Y	Y	N	60	2P_1+2C
T_Y8___U8V8_N420 (YUV420 semi-planar)	e.g., NV21	Y	Y	Y	Y	N	61	2P_1+2C
T_Y8___V8U8_N420 (YUV420 semi-planar)	e.g., NV12	Y	Y	Y	Y	Y*	62	2P_1+2C
T_Y8___U8___V8_N444 (YUV444 planar)	e.g., YV24	Y	Y	Y	Y	N	63	3P_1+1+1C

Pixel Format	FOURCC	Input Support	Output Support	Rotation	Scaling	High Quality Deinterlacing	New index	Color_Conversion_Mode
T_Y8___U8___V8_N422 (YUV422 planar)	e.g., YV16	Y	Y	Y	Y	N	64	3P_1+1+1C
T_Y8___U8___V8_N422R (YUV422R planar)		Y	Y	Y	Y	N	65	3P_1+1+1C
T_Y8___U8___V8_N420 (YUV420 planar)	e.g., YV12	Y	Y	Y	Y	Y	66	3P_1+1+1C

*Currently, DiSi1 deinterlacing only works on NV24 surfaces and field-based variants of NV12, YV12, YUY2, and UYVY formats (with each field stored in a separate surface).

In the color format, the notation called “A:B:C” notation is used to describe how often U and V are sampled relative to Y.

444 (4:4:4) means no downsampling of the chroma channels.

422 (4:2:2) means 2:1 horizontal downsampling, with no vertical downsampling. Every scan line contains four Y samples for every two U and V samples.

420 (4:2:0) means 2:1 horizontal downsampling, with 2:1 vertical downsampling.

R refers to ‘rotated’. 422 by default means horizontal downsampling; while 422R means vertical downsampling. Instead of the chroma being shared between 2 horizontally adjacent luma components, it is shared between 2 vertically adjacent luma components.

The VIC does not support 24-bit color formats, either RGB or YUV.

13.3.10 Scaling and Filtering

The VIC contains Y and X-scaler units that together allow 5- or 10-tap polyphase filtering and scaling of the input surface. The hardware can be programmed with a set of filter coefficients for each filter-type (Detail/Noise/Normal), phase (0,1/32,2/32, .. 32/32), scale ratio (1:1, 2:1, 4:1, 8:1, 16:1), filter_length (5-tap, 10-tap) and stream (sub-picture or normal). Based on the actual scale-ratio, phase, etc., the hardware can then interpolate between these programmed values for use in the actual filters.

There are four filter types (5-tap non-substream, 5-tap substream, 10-tap non-substream, 10-tap substream). Five tables are needed for each coefficient type as different tables are needed for different scale down ratio of 1:1, 2:1, 4:1, 8:1 and 16:1[1]. Thus 20 tables are needed with a maximum downscale ratio of 16:1.

Each table has 16X4 memory entries (for the 5-tap filter) or 16X9 memory entries (for 10-tap filter). Each memory entry has three 10-bit base coefficients (for Detail/Noise/Default) that will be feed into a LERP3 simultaneously to interpolate based on different noise and detail weight. The table has 16 columns, and each column stores all the coefficients for phase 0/32, 1/32, 2/32, 3/32, 4/32, 5/32, 6/32, 7/32, 8/32, 9/32, 10/32, 11/32, 12/32, 13/32, 14/32 and 15/32, with the exception that the first column stores coefficients for both 0/32 and 16/32.

Each column stores four sets of base coefficients for the 5-tap filter because the last coefficient can be derived from the remaining four. phase 0/32 and 16/32 are special cases in that only two sets of base coefficients are needed for each of them in the 5-tap case,only four sets of base coefficients are needed for phase 0/32 in the 10-tap case, and five sets of base coefficients are needed for phase 16/32 in the 10-tap case. Therefore, phase 0/32 and 16/32 are folded into the same column. Due to symmetry, the base coefficients for phases 17 through 31 do not need to be stored.

[1] 1:1 ratio contains identify filter that can be used for all scale up cases. Different scale down ratio requires different coefficients since the filter is a low pass filter the bandwidth of which is related to the scale ratio. Coefficients for 2:1, 4:1, 8:1 and 16:1 are stored to interpolate any downscale ratios.

13.3.10.1 Panoramic Scaling

The VIC also implements a “panoramic” scaling mode which can be used to convert images between different aspect ratios while avoiding any letterboxing. This mode essentially implements a non-linear scaling in the horizontal direction, with the scale ratio changing as we move away from the center of the image.

Panoramic scaling has an additional parameter *p* ([-1, 1]) that defines the magnitude of the second order term in a polynomial. The polynomial parameters for scaling can then be defined as follows.

The original scaling factor *f* is changed based on the location *x* ([-1, 1]) within the destination rectangle.

$$scaling\_factor_x = f*(a+3bx^2) \text{ with } a=\begin{cases}1-\frac{1}{2}p & p<0\\ 1-p & p\geq 0\end{cases} \text{ and } b=\begin{cases}\frac{1}{2}p & p<0\\ p & p\geq 0\end{cases}$$

The source sample position *x'* is therefore calculated as:

$$x' = f*(ax+bx^3) = a'x+b'x^3 \text{ with } a' = fa \text{ and } b' = fb$$

The new scaling factor can be calculated incrementally using 2 counters.

Panoramic scaling for 4:3 to 16:9 conversion should have $p=-\frac{2}{3}$, while 16:9 to 4:3 conversion should have $p=\frac{1}{4}$ for 1:1 scaling in the center.

13.3.11 Pixel Format Conversion

Color conversion and proc-amp are implemented using a matrix multiplication on every pixel. The matrix only changes when the surfaces changes.

Current chain of operations:

- Proc-amp + Output CC (4x3 matrix)
- Clamping (optional soft clamping, 2 control entry)

The 4x3 matrix values are specified in the ColorConversionMatrixStruct as S12.8 values. In addition, the result of the matrix multiplication will be right shifted by r_shift. To allow for highest accuracy r_shift should always be as high as possible without losing any range in the other coefficients. Note that the constant offsets (c03, c13, c23) are S12.8 and are not affected by r_shift.

$$\begin{pmatrix}out0\\ out1\\ out2\end{pmatrix} = \begin{pmatrix}c00 & c01 & c02 & c03\\ c10 & c11 & c12 & c13\\ c20 & c21 & c22 & c23\end{pmatrix}\begin{pmatrix}in0 \gg r\_shift\\ in1 \gg r\_shift\\ in2 \gg r\_shift\\ 1\end{pmatrix}$$

Soft clamping defines a piecewise linear function consisting of three pieces. The control entries define the area of soft clamping. For example, the following equation applies for the values *a* and *b*:

$$v' = \begin{cases}0 & v<-a\\ \frac{v+a}{2} & -a\leq v<a\\ v & a\leq v<b\\ \frac{v+b}{2} & b\leq v<2-b\\ 1 & 2-b\leq v\end{cases}$$

When converting between pixel formats with components of varying bit width, bit replication is used for expanding component bit widths, and truncation is used for reducing component bit widths.

Dithering and undithering are not supported.

Data Size Conversion

When expanding the number of bits in a component (e.g., from RGB565 to RGB888), expansion should use bit replication of the MSBs of the original value to fill in the LSBs of the expanded bit length value. For example, if converting from a 5-bit value to an 8-bit value, the first 3 MSBs of the source 5-bit value should be concatenated as the 3 new LSBs to the original 5-bit value to produce the new expanded 8-bit value.

For example: 5-bit R5=**110**00'b expanded to 8 bit R8=**110**00*110*'b

When reducing the number of bits in a component (for example, from RGB888 to RGB565), the LSBs of the old value should be truncated to arrive at the shortened value.

For example: 8-bit R8=11000*000*'b reduced to 5 bit R5=11000'b

Replication followed by truncation will ensure that the truncated value will equal the original pre-replicated operand, i.e., RGB565→RGB888 expansion followed immediately by RGB888→RGB565 reduction will result in the same original value.

Luminance Conversion

Conversion from luminance-only color formats (e.g., L8/Y8/YUV100, A8L8) to and from RGB or YUV formats should be done as outlined below.

- Luminance-only to YUV

  L→Y, 128→U, 128→V
- Luminance-only to RGB

  L→R, L→G, L→B
- YUV to Luminance-only

  Y→L
- Clamp L between 0.0 and 1.0
- RGB to Luminance-only

  (R*5 + G*9 + B*2)/16 → L
- Clamp L between 0.0 and 1.0

Adding and Removing Alpha

If converting from a pixel format with alpha to a pixel format with no alpha, the alpha value should be dropped.

If converting from a pixel format with no alpha to a pixel format with alpha, the alpha value in the destination pixel format should be set to 1.0.

13.3.12 Luma Keying

Coefficient values to compute the Luma value of the pixel from the pixel components can be specified in the ColorConversionLumaAlphaStruct, along with the upper and lower luma key values. Any pixel that has a luma that falls within the luma key range will have its alpha value zeroed out so that it can be processed appropriately in the blending stage.

13.3.13 Blender

The blender interface allows for symmetric blend modes between the VIC and Display units.

The VIC blender supports the following different blending modes:

- DXVAHD_ALPHA_FILL_MODE_OPAQUE
- DXVAHD_ALPHA_FILL_MODE_BACKGROUND
- DXVAHD_ALPHA_FILL_MODE_DESTINATION
- DXVAHD_ALPHA_FILL_MODE_SOURCE_STREAM
- DXVAHD_ALPHA_FILL_MODE_COMPOSITED
- DXVAHD_ALPHA_FILL_MODE_SOURCE_ALPHA

All modes other than _ALPHA_FILL_MODE_COMPOSITED are as defined in the Microsoft DXVA spec and are carried over from VIC 1.0. The new mode that allows the VIC and Display blenders to match each other is enabled by setting the AlphaFillModein the VIC Config Structure to _ALPHA_FILL_MODE_COMPOSITED. When the AlphaFillMode is set to anything other than _ALPHA_FILL_MODE_COMPOSITED, the srcFact and dstFact parameters specified below are ignored and do not affect the processing.

The parameterizable blender interface allows a variety of blend modes, including the following:

- Per-Pixel Non-Premultiplied Alpha Blend

  output = src*src_alpha + dst*(1-src_alpha)
- Per Pixel Source Premultiplied Alpha Blend

  output = src + dst*(1-src_alpha)
- Constant-Alpha Blend

  output = src*const_alpha + dst*(1-const_alpha)
- Per-Pixel Non-Premultiplied Alpha Blend with constant blend

  output = src*src_alpha*const_alpha + dst*(1-src_alpha*const_alpha)
- Per-Pixel Premultiplied Alpha Blend with constant blend

  output = src*const_alpha + dst*(1-src_alpha*const_alpha)

Note that ColorKeySelect should always be set to disabled.

The blend data path will require the color key comparison match result, and per-slot inputs from the config structure to formulate the blend equation.

srcFactC, srcFactA, dstFactC, and dstFactA multiplicand inputs are determined by the config structure programming and color key comparison results. Note that VIC's internal pixel pipeline represents each pixel component as 10 bits, and as a result, constants from the config struct also require 10-bit representation.

Table 40 : Per-Slot Blend Configuration

Blend Spec Name	Register Name	Description
K1	AlphaK1	10-bit constant alpha value
K2	AlphaK2	10-bit constant alpha value
srcFactC_Match_Select	srcFactCMatchSelect	3-bit enum which sets srcFactC: K1: AlphaK1 K1_TIMES_DST: AlphaK1*dstA NEG_K1_TIMES_DST: 1.0-(AlphaK1*dstA) K1_TIMES_SRC: AlphaK1*srcA ZERO: 0
dstFactC_Match_Select	dstFactCMatchSelect	3-bit enum which sets dstFactC: K1: AlphaK1 K2: AlphaK2 K1_TIMES_DST: AlphaK1*dstA

Blend Spec Name	Register Name	Description
		NEG_K1_TIMES_DST: 1.0-(AlphaK1*dstA) NEG_K1_TIMES_SRC: 1.0-(AlphaK1*srcA) ZERO: 0 ONE: 1.0
srcFactA_Match_Select	srcFactAMatchSelect	3-bit enum which sets srcFactA: K1: AlphaK1 K2: AlphaK2 ZERO: 0 NEG_K1_TIMES_DST: 1.0-(AlphaK1*dstA)
dstFactA_Match_Select	dstFactAMatchSelect	3-bit enum which sets dstFactA: K2: AlphaK2 NEG_K1_TIMES_SRC: 1.0-(AlphaK1*srcA) ZERO: 0 ONE: 1.0
UseK3	UseOverrideR	1-bit Boolean to override srcR with constant values K3R
UseK3	UseOverrideG	1-bit Boolean to override srcG with constant values K3G
UseK3	UseOverrideB	1-bit Boolean to override srcB with constant values K3B
UseK3	UseOverrideA	1-bit Boolean to override srcA with constant values K3A
K3R	OverrideR	10-bit source override R value
K3G	OverrideG	10-bit source override G value
K3B	OverrideB	10-bit source override B value
K3A	OverrideA	10-bit source override A value
MaskR	MaskR	1-bit Boolean to override outputR to equal dstR
MaskG	MaskG	1-bit Boolean to override outputG to equal dstG
MaskB	MaskB	1-bit Boolean to override outputB to equal dstB
MaskA	MaskA	1-bit Boolean to override outputA to equal dstA

Given the source and destination factor settings based on config structure inputs and color match results, the per-pixel blend equations are as follows, where srcR, srcG, srcB, srcA, dstR, dstG, dstB, and dstA represent the incoming R, G, B, A components of the incoming source and already present destination pixels in the SFMem buffer (blended results of background and slot[0] through to slot[n-1] for input slot[n]).

```
outputR = (srcFactC * srcR) + (dstFactC * dstR)
outputG = (srcFactC * srcG) + (dstFactC * dstG)
outputB = (srcFactC * srcB) + (dstFactC * dstB)
outputA = (srcFactA * srcA) + (dstFactA * dstA)
```

If the ColorKeySelect is set to DISABLED, then srcFactC, srcFactA, dstFactC, and dstFactA will be programmed by srcFactCMatchSelect, srcFactCMatchSelect, srcFactCMatchSelect, and srcFactCMatchSelect, respectively.

If the ColorKeySelect is set to ENABLED, then srcFactC, srcFactA, dstFactC, and dstFactA are programmed by srcFactCMatchSelect, srcFactCMatchSelect, srcFactCMatchSelect, and srcFactCMatchSelect, respectively, if the destination color key match result returns true, and programmed by srcFactCNoMatchSelect, srcFactCNoMatchSelect, srcFactCNoMatchSelect, and srcFactCNoMatchSelect, respectively, if the destination color key match result returns false.

UseK3 allows the incoming source components to be overrided by constant color components K3R, K3G, K3B, K3A.

MaskR, MaskG, MaskB, and MaskA Boolean settings provide additional per-component mask bits to allow pass-through of the destination pixel components to the output pixel

There is no post-blend color space conversion available. As a result, blending should be done in the destination color space. If the destination color space is RGB, all input slots should be converted to RGB color space before blending.

A programmable background color will still be set to fill the target rectangle background via the Blending0Struct, as in VIC1.0.

13.3.14 Subpixel Source Rectangles

Source rectangles can be specified to a sub-pixel resolution. This feature can be used to smoothly zoom into a portion of an image in real-time, for example, when using the zoom button on a camera. Specifying source and destination rectangles to a pixel resolution quantizes the allowed zoom factor, and hence can cause the zoom to feel jerky. Allowing the source rectangles to be specified to a sub-pixel resolution avoids this problem and makes the zoom smooth.

The config structure is therefore modified to make the source rectangle coordinates U14.16 numbers. Once we have the (decimal) source rectangles, we can calculate adjust the starting phase of the scaling filters accordingly so as to achieve the sub-pixel shift/scale required.

Note that all source coordinates (left, top, right, bottom) include the bounding pixels. For instance: (0,0)-(0,0) contains 1 pixel, while (0,0)-(1-1) contains a square of 2x2 pixels. This means that rectangles which are less than 1 pixel wide or tall cannot be used. Also, specifying a rectangle of (0.0,0.0)-(0.0,0.1) results in a source rectangle that is 1 pixel wide and 1.1 pixel high, which is not immediately apparent.

13.3.15 Maximum Surface and Rectangle Size Support

The VIC supports surface and rectangle sizes up to 16384x16384 pixels.

This size affects all input and output surfaces parameters as well as all fetch calculations related to these parameters:

- Per-input slot and output surface width/height, luma width/height, chroma width/height
- Per-input slot source and destination rectangle dimensions
- Clear rectangle dimensions
- Output target rectangle dimensions

13.4 Programming Guidelines

13.4.1 Class Operation

The operation of this class can be summarized as follows:

1. The driver should set up input and output context DMAs as specified here.
2. After setting up the surfaces, the driver should send the parameter methods which specify:
   - Control Configuration
   - Configuration structure offset
   - Palette buffer offset
   - History buffer offset
   - Input/Output/Intermediate surface offsets
   - Context ID methods
   - Fetch Control Engine (FCE) microcode offset
3. When Execute() method is sent, the Falcon program will be initiated.

The compositor operation requires the following buffers. These buffers are explained in more detail in the following sections.

Buffers	I/O	Producer	Description/Restrictions
Config Buffer	I	Driver	Configuration parameters
Palette Buffer	I	Driver	Palette table
History Buffer	I/O	App	Histogram and Cadence parameters
Input surfaces	I	Driver/VIC	Input Surfaces
Noise reduced surfaces	I/O	VIC	Noise reduced version of input
Motion map surfaces	I/O	VIC	Motion map of input
FCE Microcode buffer	I	Driver	FCE microcode

13.4.1.1 Restrictions

- Maximum slots (MAX_SLOTS) supported are 5.
- Maximum downscale ratio is 16:1.
- When enabling panoramic scaling, the maximum downscale ratio is 7:1. In addition, the minimum destination rectangle size is 4x4 pixels.
- Minimum source rectangle size is 4 pixels x 4 pixels.
- The only pixel formats supported in video pre-process operations (IVTC, TNR, DiSi1 deinterlacing) are NV12, YV12, YUY2, and UYVY.

13.4.2 Application Memory Structures

13.4.2.1 Enumerates and Constants

Below are the enumerators used in the configuration structures passed by the driver:

- DXVAHD_FRAME_FORMAT
- PIXEL_FORMAT
- DXVAHD_DEINTERLACE_MODE_PRIVATE
- DXVAHD_ALPHA_FILL_MODE
- BLK_KIND
- BLEND_SRCFACTC
- BLEND_DSTFACTC
- BLEND_SRCFACTA
- BLEND_DSTFACTA

13.4.3 Config Structure

The Config structure is read from the frame buffer by the surface cache unit and is stored in small memories inside each subunit. The entire ConfigStruct is the concatenation of the Surface Cache, Surface List, Color Conversion, Blending and Fetch Control structs (in this order). The structs are read in 128-bit units from memory and sent to the according subunits memory with the correct memory address by the Surface Cache.

The config structure is broken up into smaller structs that are used depending on the active slots and their content. Each structure is a multiple of 128 bits in size, and the start of the struct needs to be aligned to a 256 byte boundary.

Note: NV12 and NV24 share the same pixel format but can be distinguished by the fact that NV24 is a field based format (i.e., every surface contains a single field) whereas NV12 is frame based (i.e., every surface contains an entire frame).

Note: For highest quality, filter override mode as defined in FetchControlCoeffStruct should always be used.

The following table defines the smaller structs that comprise ConfigStruct.

Table 41: ConfigStruct Substructures

Name	Data Type	Offset	Notes
surfaceCache0Struct	SurfaceCache0Struct (128 bits)	0	
surfaceList0Struct	SurfaceList0Struct (128 bits)	128	
surfaceListClearRectStruct[4]	SurfaceListClearRectStruct (128 bits)	256 + (i * 128)	Up to eight clear rectangles supported. Each SurfaceListClearRectStruct specifies two rectangles
surfaceListSurfaceStruct[5]	SurfaceListSurfaceStruct (384 bits)	768 + (i * 384)	
colorConversionLumaAlphaStruct[5]	ColorConversionLumaAlphaStruct (128 bits)	2688 + (i * 128)	
colorConversionMatrixStruct[5]	ColorConversionMatrixStruct (256 bits)	3328 + (i * 256)	Conversion matrix for brightness, hue and saturation values
colorConversionClampStruct[5]	ColorConversionClampStruct (128 bits)	4608 + (i * 128)	
blending0Struct	Blending0Struct (256 bits)	5248	
blendingSurfaceStruct[5]	BlendingSurfaceStruct (128 bits)	5504 + (i * 128)	
fetchControl0Struct	FetchControl0Struct (512 bits)	6144	
fetchControlCoeffStruct[520]	FetchControlCoeffStruct (128 bits)	6656 + (i * 128)	520 coefficient structs

13.4.3.1 SurfaceCache0Struct

This structure contains enable bits for each slot, which need to be set as described below.

- Noise reduction requires a forward and a backward reference field for interlaced content and a backward reference for progressive content (see the “Method Naming and Programming” section about how to set surfaces). Both require a noise reduced surface (field for interlaced, frame for progressive).
- MotionMap calculation is required for DiSi1/DiNewBob and also needs a forward and a backward reference. Behavior is not defined for progressive streams so enabling it needs to be flagged as an error.
- Cadence Detection enables artifact and weave counts. It also enables Falcon code to force deinterlace mode to weave if a cadence was detected.

Table 42: SurfaceCache0Struct Details

Name	Data Type	Offset	Notes
DeNoise0	fixed<0,1,0> (1 bit)	0	Enable bit for noise reduction
CadenceDetect0	fixed<0,1,0> (1 bit)	1	Cadence detection enable bit
MotionMap0	fixed<0,1,0> (1 bit)	2	Motion map enable bit for each slot. Required for DiSi1/DiNewBob
MedianFilter0	fixed<0,1,0> (1 bit)	3	Do not enable Median filtering
DeNoise1	fixed<0,1,0> (1 bit)	4	

Name	Data Type	Offset	Notes
CadenceDetect1	fixed<0,1,0> (1 bit)	5	
MotionMap1	fixed<0,1,0> (1 bit)	6	
MedianFilter1	fixed<0,1,0> (1 bit)	7	
DeNoise2	fixed<0,1,0> (1 bit)	8	
CadenceDetect2	fixed<0,1,0> (1 bit)	9	
MotionMap2	fixed<0,1,0> (1 bit)	10	
MedianFilter2	fixed<0,1,0> (1 bit)	11	
DeNoise3	fixed<0,1,0> (1 bit)	12	
CadenceDetect3	fixed<0,1,0> (1 bit)	13	
MotionMap3	fixed<0,1,0> (1 bit)	14	
MedianFilter3	fixed<0,1,0> (1 bit)	15	
DeNoise4	fixed<0,1,0> (1 bit)	16	
CadenceDetect4	fixed<0,1,0> (1 bit)	17	
MotionMap4	fixed<0,1,0> (1 bit)	18	
MedianFilter4	fixed<0,1,0> (1 bit)	19	
IsEven0	fixed<0,1,0> (1 bit)	20	Select if current field is even (used for cadence detection)
IsEven1	fixed<0,1,0> (1 bit)	21	
IsEven2	fixed<0,1,0> (1 bit)	22	
IsEven3	fixed<0,1,0> (1 bit)	23	
IsEven4	fixed<0,1,0> (1 bit)	24	
MMapCombine0	fixed<0,1,0> (1 bit)	25	Enable bit for combine motion map for each slot. This is required for DiSi1/DiNewBob. This requires MotionMap being enabled and also required a prevMotionMap surface (see “Method Naming and Programming”). Behavior without MotionMap enabled is undefined and has to be flagged as an error.
MMapCombine1	fixed<0,1,0> (1 bit)	26	
MMapCombine2	fixed<0,1,0> (1 bit)	27	
MMapCombine3	fixed<0,1,0> (1 bit)	28	
MMapCombine4	fixed<0,1,0> (1 bit)	29	
reserved0	fixed<0,2,0> (2 bits)	30	
PixelFormat0	fixed<0,7,0> (7 bits)	32	Pixel format for each stream (PIXEL_FORMAT) Only NV12, YUY2 and UYVY allow for motion buffer calculation and DiSi1/DiNewBob deinterlacing and noise reduction.
reserved1	fixed<0,1,0> (1 bit)	39	

Name	Data Type	Offset	Notes
PixelFormat1	fixed<0,7,0> (7 bits)	40	
reserved2	fixed<0,1,0> (1 bit)	47	
PixelFormat2	fixed<0,7,0> (7 bits)	48	
reserved3	fixed<0,1,0> (1 bit)	55	
PixelFormat3	fixed<0,7,0> (7 bits)	56	
reserved4	fixed<0,1,0> (1 bit)	63	
PixelFormat4	fixed<0,7,0> (7 bits)	64	
reserved5	fixed<0,1,0> (1 bit)	71	
reserved6	fixed<0,24,0> (24 bits)	72	
PPMotion0	fixed<0,1,0> (1 bit)	96	Enable bit for previous motion calculation for each slot. PPMotion is obsolete and should not be enabled.
PPMotion1	fixed<0,1,0> (1 bit)	97	
PPMotion2	fixed<0,1,0> (1 bit)	98	
PPMotion3	fixed<0,1,0> (1 bit)	99	
PPMotion4	fixed<0,1,0> (1 bit)	100	
reserved7	fixed<0,3,0> (3 bits)	101	
ChromaEven0	fixed<0,1,0> (1 bit)	104	Enable if chroma of current field is even
ChromaEven1	fixed<0,1,0> (1 bit)	105	
ChromaEven2	fixed<0,1,0> (1 bit)	106	
ChromaEven3	fixed<0,1,0> (1 bit)	107	
ChromaEven4	fixed<0,1,0> (1 bit)	108	
reserved8	fixed<0,3,0> (3 bits)	109	
AdvancedDenoise0	fixed<0,1,0> (1 bit)	112	Enable the advanced denoising algorithm (TNR2) for each slot.
AdvancedDenoise1	fixed<0,1,0> (1 bit)	113	
AdvancedDenoise2	fixed<0,1,0> (1 bit)	114	
AdvancedDenoise3	fixed<0,1,0> (1 bit)	115	
AdvancedDenoise4	fixed<0,1,0> (1 bit)	116	
reserved9	fixed<0,3,0> (3 bits)	117	
reserved10	fixed<0,8,0> (8 bits)	120	

13.4.3.2 SurfaceList0Struct

The per-slot clear rectangle masks enable or disable each of the 8 clear rectangles defined in the SurfaceListClearRectStruct for that slot. This structure also specifies the output flip enables and Target rectangle.

Table 43: SurfaceList0Struct Details

Name	Data Type	Offset	Notes
ClearRectMask0	fixed<0,8,0> (8 bits)	0	Clear rectangle mask for slot 0
ClearRectMask1	fixed<0,8,0> (8 bits)	8	Clear rectangle mask for slot 1
ClearRectMask2	fixed<0,8,0> (8 bits)	16	Clear rectangle mask for slot 2
ClearRectMask3	fixed<0,8,0> (8 bits)	24	Clear rectangle mask for slot 3
ClearRectMask4	fixed<0,8,0> (8 bits)	32	Clear rectangle mask for slot 4
reserved0	fixed<0,22,0> (22 bits)	40	
OutputFlipX	fixed<0,1,0> (1 bit)	62	Flip output image horizontally
OutputFlipY	fixed<0,1,0> (1 bit)	63	Flip output image vertically
TargetRectLeft	fixed<0,14,0> (14 bits)	64	Target rectangle. Restricts the output to a certain rectangle inside the output surface. Pixels outside of this area are guaranteed to remain unmodified.
reserved1	fixed<0,2,0> (2 bits)	78	
TargetRectRight	fixed<0,14,0> (14 bits)	80	
reserved2	fixed<0,2,0> (2 bits)	94	
TargetRectTop	fixed<0,14,0> (14 bits)	96	
reserved3	fixed<0,2,0> (2 bits)	110	
TargetRectBottom	fixed<0,14,0> (14 bits)	112	
reserved4	fixed<0,2,0> (2 bits)	126	

13.4.3.3 SurfaceListClearRectStruct[4]

The SurfaceListClearRectStruct structures together define 8 clear rectangles. These rectangles are enabled or disabled for each slot using the fields in SurfaceList0Struct.

Table 44: SurfaceListClearRectStruct[4] Details

Name	Data Type	Offset	Notes
ClearRect0Left	fixed<0,14,0> (14 bits)	0	First clear rectangle of 128-bit chunk
reserved0	fixed<0,2,0> (2 bits)	14	
ClearRect0Right	fixed<0,14,0> (14 bits)	16	
reserved1	fixed<0,2,0> (2 bits)	30	

Name	Data Type	Offset	Notes
ClearRect0Top	fixed<0,14,0> (14 bits)	32	
reserved2	fixed<0,2,0> (2 bits)	46	
ClearRect0Bottom	fixed<0,14,0> (14 bits)	48	
reserved3	fixed<0,2,0> (2 bits)	62	
ClearRect1Left	fixed<0,14,0> (14 bits)	64	Second clear rectangle of 128-bit chunk
reserved4	fixed<0,2,0> (2 bits)	78	
ClearRect1Right	fixed<0,14,0> (14 bits)	80	
reserved5	fixed<0,2,0> (2 bits)	94	
ClearRect1Top	fixed<0,14,0> (14 bits)	96	
reserved6	fixed<0,2,0> (2 bits)	110	
ClearRect1Bottom	fixed<0,14,0> (14 bits)	112	
reserved7	fixed<0,2,0> (2 bits)	126	

13.4.3.4 SurfaceListSurfaceStruct[5]

Surface parameters for each slot.

Table 45: SurfaceListSurfaceStruct[5] Details

Name	Data Type	Offset	Notes
Enable	fixed<0,1,0> (1 bit)	0	Enable or disable bit for this slot
FrameFormat	fixed<0,4,0> (4 bits)	1	Frame format of the frame (DXVAHD_FRAME_FORMAT)
PixelFormat	fixed<0,7,0> (7 bits)	5	Pixel format for each stream (PIXEL_FORMAT)
reserved0	fixed<0,2,0> (2 bits)	12	
ChromaLocHoriz	fixed<0,2,0> (2 bits)	14	Horizontal chroma location 0: Co-located with even luma 1: In between even and odd luma 2: Co-located with odd luma 3: Between odd and next even luma
ChromaLocVert	fixed<0,2,0> (2 bits)	16	Vertical chroma location 0: Co-located with even luma 1: In between even and odd luma 2: Co-located with odd luma 3: Between odd and next even luma
Panoramic	fixed<0,12,0> (12 bits)	18	Panoramic scaling parameter. For details on this parameter, refer to the Panoramic Scaling subsection.
reserved1	fixed<0,4,0> (4 bits)	30	
SurfaceWidth	fixed<0,14,0> (14 bits)	34	Width of surface minus 1. Any pixel data outside of this will not be used inside the VIC but might still be read.

Name	Data Type	Offset	Notes
reserved2	fixed<0,1,0> (1 bit)	48	
SurfaceHeight	fixed<0,14,0> (14 bits)	49	Height of surface minus 1. Any pixel data outside of this will not be used inside the VIC but might still be read.
reserved3	fixed<0,1,0> (1 bit)	63	
LumaWidth	fixed<0,14,0> (14 bits)	64	Padded luma width of surface minus 1. Any pixel data outside of this will not be read.
reserved4	fixed<0,1,0> (1 bit)	78	
LumaHeight	fixed<0,14,0> (14 bits)	79	Padded luma height of surface minus 1. Any pixel data outside of this will not be read.
reserved5	fixed<0,1,0> (1 bit)	93	
ChromaWidth	fixed<0,14,0> (14 bits)	94	Padded chroma width of surface minus 1. Any pixel data outside of this will not be read. This value is not required for pixel interleaved surfaces such as ARGB.
reserved6	fixed<0,1,0> (1 bit)	108	
ChromaHeight	fixed<0,14,0> (14 bits)	109	Padded chroma height of surface minus 1. Any pixel data outside of this will not be read. This value is not required for pixel interleaved surfaces such as ARGB
reserved7	fixed<0,1,0> (1 bit)	123	
CacheWidth	fixed<0,3,0> (3 bits)	124	Number of horizontal bytes per surface-cache cache-line. Each 256B cache line can store a source region of size as enumerated below. 0: 16Bx16 (BL16Bx2) 1: 32Bx8 (BL16Bx2) 2: 64Bx4 (BL16Bx2, PL) 3: 128Bx2 (BL16Bx2, PL) 4: 256Bx1 (PL)
reserved8	fixed<0,1,0> (1 bit)	127	
FilterLengthY	fixed<0,2,0> (2 bits)	128	Filter length for each stream: 0: 1-tap (nearest neighbor) 1: 2-tap (bi-linear) 2: 5-tap 3: 10-tap
FilterLengthX	fixed<0,2,0> (2 bits)	130	
DetailFltClamp	fixed<0,6,0> (6 bits)	132	
reserved9	fixed<0,2,0> (2 bits)	138	
LightLevel	fixed<0,4,0> (4 bits)	140	Describes the level of lighting present when the input image was captured. This parameter is used along with the AdvancedDenoise bits to determine the exact denoising algorithm to be applied for noise reduction.
reserved10	fixed<0,4,0> (4 bits)	144	
reserved11	fixed<0,8,0> (8 bits)	148	
reserved12	fixed<0,32,0> (32 bits)	156	
BlkKind	fixed<0,4,0> (4 bits)	188	Block-linear kind (BLK_KIND)

Name	Data Type	Offset	Notes
DestRectLeft	fixed<0,14,0> (14 bits)	192	The destination rectangle defines the region of the output surface that is affected by this slot. Together with the source rectangle it also defines the scaling rations. Any pixel outside of this region will not be affected by this input stream. For any non-4:4:4 format, all corners need to fall on a multiple of 2 (Right and Bottom are minus 1 encoded though).
reserved13	fixed<0,1,0> (1 bit)	206	
DestRectRight	fixed<0,14,0> (14 bits)	207	
reserved14	fixed<0,1,0> (1 bit)	221	
DestRectTop	fixed<0,14,0> (14 bits)	222	
reserved15	fixed<0,1,0> (1 bit)	236	
DestRectBottom	fixed<0,14,0> (14 bits)	237	
reserved16	fixed<0,1,0> (1 bit)	251	
BlkHeight	fixed<0,4,0> (4 bits)	252	Block-linear height (in gobs, log2)
SourceRectLeft	fixed<0,30,0> (30 bits)	256	The source rectangle defines the region of pixels that will be read from the source surface. Any pixel data outside of this will not be used inside the VIC but might still be read. The source rectangle coordinates are specified in a U14.16 format, allowing fractional coordinates to be specified. The source rectangle should lie entirely within the input surface for the slot; that is, negative values, or values greater than the size of the surface, are illegal.
reserved17	fixed<0,2,0> (2 bits)	286	
SourceRectRight	fixed<0,30,0> (30 bits)	288	
reserved18	fixed<0,2,0> (2 bits)	318	
SourceRectTop	fixed<0,30,0> (30 bits)	320	
reserved19	fixed<0,2,0> (2 bits)	350	
SourceRectBottom	fixed<0,30,0> (30 bits)	352	
reserved20	fixed<0,2,0> (2 bits)	382	

13.4.3.5 ColorConversionLumaAlphaStruct[5]

Enables luma keying and plane alpha parameters.

Table 46: ColorConversionLumaAlphaStruct[5] Details

Name	Data Type	Offset	Notes
l0	fixed<0,20,0> (20 bits)	0	Matrix entry (0) of 4x1 luma conversion matrix in S12.8 format
l1	fixed<0,20,0> (20 bits)	20	Matrix entry (1) of 4x1 luma conversion matrix in S12.8 format
l2	fixed<0,20,0> (20 bits)	40	Matrix entry (2) of 4x1 luma conversion matrix in S12.8 format

Name	Data Type	Offset	Notes
r_shift	fixed<0,4,0> (4 bits)	60	The result of the matrix multiplication is right shifted by r_shift. To allow for highest accuracy r_shift should always be as high as possible without losing any range in the other coefficients.
l3	fixed<0,20,0> (20 bits)	64	Matrix entry (3) of 4x1 luma conversion matrix in S12.8 format. Is not affected by r_shift.
PlanarAlpha	fixed<0,10,0> (10 bits)	84	10-bit planar alpha value. This planar alpha is multiplied with the alpha value coming from the stream. In case of palettized alpha formats (like AI44/A8P8/AI88), the stream alpha value is the alpha value from the surface multiplied with the alpha value from the palette.
ConstantAlpha	fixed<0,1,0> (1 bit)	94	If true planar alpha value is used instead of stream alpha, if false, stream alpha is multiplied with planar alpha. Constant alpha has to be set for all surfaces not containing any alpha data (like XRGB/NV12/YUY2/UYVY/YV12).
ClipEnabled	fixed<0,1,0> (1 bit)	95	Enables clip against negative values after gamut matrix
LumaKeyLower	fixed<0,10,0> (10 bits)	96	Lower luma key value
reserved6	fixed<0,3,0> (3 bits)	106	
StereoInterleave	fixed<0,3,0> (3 bits)	109	Enables pixel interleave for auto-stereoscopic panels (STEREO_INTERLEAVE)
LumaKeyUpper	fixed<0,10,0> (10 bits)	112	Upper luma key value
reserved7	fixed<0,2,0> (2 bits)	122	
reserved8	fixed<0,1,0> (1 bit)	124	
LumaKeyEnabled	fixed<0,1,0> (1 bit)	125	Luma key enable
reserved9	fixed<0,2,0> (2 bits)	126	

13.4.3.6 ColorConversionMatrixStruct[5]

The matrices defined in these structures specify the color space conversion between input and output pixel formats, if any.

Table 47: ColorConversionMatrixStruct[5] Details

Name	Data Type	Offset	Notes
c00	fixed<0,20,0> (20 bits)	0	Matrix entry (0,0) of 4x3 color conversion matrix. Precision and right shift are the same as for the luma vector.
c10	fixed<0,20,0> (20 bits)	20	Matrix entry (1,0) of 4x3 color conversion matrix
c20	fixed<0,20,0> (20 bits)	40	Matrix entry (2,0) of 4x3 color conversion matrix
r_shift	fixed<0,4,0> (4 bits)	60	Right shift value for matrix
c01	fixed<0,20,0> (20 bits)	64	Matrix entry (0,1) of 4x3 color conversion matrix
c11	fixed<0,20,0> (20 bits)	84	Matrix entry (1,1) of 4x3 color conversion matrix
c21	fixed<0,20,0> (20 bits)	104	Matrix entry (2,1) of 4x3 color conversion matrix
reserved0	fixed<0,4,0> (4 bits)	124	

Name	Data Type	Offset	Notes
c02	fixed<0,20,0> (20 bits)	128	Matrix entry (0,2) of 4x3 color conversion matrix
c12	fixed<0,20,0> (20 bits)	148	Matrix entry (1,2) of 4x3 color conversion matrix
c22	fixed<0,20,0> (20 bits)	168	Matrix entry (2,2) of 4x3 color conversion matrix
reserved1	fixed<0,4,0> (4 bits)	188	
c03	fixed<0,20,0> (20 bits)	192	Matrix entry (0,3) of 4x3 color conversion matrix
c13	fixed<0,20,0> (20 bits)	212	Matrix entry (1,3) of 4x3 color conversion matrix
c23	fixed<0,20,0> (20 bits)	232	Matrix entry (2,3) of 4x3 color conversion matrix
reserved2	fixed<0,4,0> (4 bits)	252	

13.4.3.7 ColorConversionClampStruct[5]

Specifies the upper and lower clamp boundaries for pixels after the matrix multiplication.

Table 48: ColorConversionClampStruct[5] Details

Name	Data Type	Offset	
low	fixed<0,10,0> (10 bits)	0	Lower soft clamping boundaries
reserved0	fixed<0,6,0> (6 bits)	10	
high	fixed<0,10,0> (10 bits)	16	Upper soft clamping boundaries
reserved1	fixed<0,6,0> (6 bits)	26	
reserved2	fixed<0,32,0> (32 bits)	32	
reserved3	fixed<0,32,0> (32 bits)	64	
reserved4	fixed<0,32,0> (32 bits)	96	

13.4.3.8 Blending0Struct

Specifies parameters related to the alpha-blending of the different slots and the output surface/rectangle parameters.

Table 49: Blending0Struct Details

Name	Data Type	Offset	Notes
PixelFormat	fixed<0,7,0> (7 bits)	0	Pixel format of the output surface (PIXEL_FORMAT)
reserved0	fixed<0,1,0> (1 bit)	7	
AlphaFillMode	fixed<0,3,0> (3 bits)	8	Alpha fill mode (DXVAHD_ALPHA_FILL_MODE)
AlphaFillSlot	fixed<0,3,0> (3 bits)	11	SlotId for when AlphaFillMode == Source stream/Source alpha
BackgroundAlpha	fixed<0,10,0> (10 bits)	14	Background color A

Name	Data Type	Offset	Notes
BackgroundR	fixed<0,10,0> (10 bits)	24	Background color R
BackgroundG	fixed<0,10,0> (10 bits)	34	Background color G
BackgroundB	fixed<0,10,0> (10 bits)	44	Background color B
ChromaLocHoriz	fixed<0,2,0> (2 bits)	54	Chroma location of the output surface (see SurfaceListSurfaceStruct)
ChromaLocVert	fixed<0,2,0> (2 bits)	56	
reserved1	fixed<0,6,0> (6 bits)	58	
LumaWidth	fixed<0,14,0> (14 bits)	64	Padded output surface luma width minus 1
reserved2	fixed<0,2,0> (2 bits)	78	
LumaHeight	fixed<0,14,0> (14 bits)	80	Padded output surface luma height minus 1
reserved3	fixed<0,2,0> (2 bits)	94	
ChromaWidth	fixed<0,14,0> (14 bits)	96	Padded output surface chroma width minus 1
reserved4	fixed<0,2,0> (2 bits)	110	
ChromaHeight	fixed<0,14,0> (14 bits)	112	Padded output surface chroma height minus 1
reserved5	fixed<0,2,0> (2 bits)	126	
TargetRectLeft	fixed<0,14,0> (14 bits)	128	Target rectangle, restricts the output of pixels to this region. For any non-4:4:4 format, all corners need to fall on a multiple of 2 (Right and Bottom are minus 1 encoded though).
reserved6	fixed<0,2,0> (2 bits)	142	
TargetRectRight	fixed<0,14,0> (14 bits)	144	
reserved7	fixed<0,2,0> (2 bits)	158	
TargetRectTop	fixed<0,14,0> (14 bits)	160	
reserved8	fixed<0,2,0> (2 bits)	174	
TargetRectBottom	fixed<0,14,0> (14 bits)	176	
reserved9	fixed<0,2,0> (2 bits)	190	
SurfaceWidth	fixed<0,14,0> (14 bits)	192	Output surface width minus 1
reserved10	fixed<0,2,0> (2 bits)	206	
SurfaceHeight	fixed<0,14,0> (14 bits)	208	Output surface height minus 1
reserved11	fixed<0,2,0> (2 bits)	222	
BlkKind	fixed<0,4,0> (4 bits)	224	The block linear kind of the output surface (BLK_KIND)
BlkHeight	fixed<0,4,0> (4 bits)	228	Block-linear height (in gobs, log2)
OutputFlipX	fixed<0,1,0> (1 bit)	232	Horizontal Flip enable

Name	Data Type	Offset	Notes
OutputFlipY	fixed<0,1,0> (1 bit)	233	Vertical Flip enable
OutputTranspose	fixed<0,1,0> (1 bit)	234	Transpose enable
reserved12	fixed<0,21,0> (21 bits)	235	

13.4.3.9 BlendingSurfaceStruct[5]

Input parameters to blend equations in DXVAHD_ALPHA_FILL_MODE_COMPOSITED alpha fill mode

- outputR = (srcFactC * srcR) + (dstFactC * dstR)
- outputG = (srcFactC * srcG) + (dstFactC * dstG)
- outputB = (srcFactC * srcB) + (dstFactC * dstB)
- outputA = (srcFactA * srcA) + (dstFactA * dstA)

Table 50: BlendingSurfaceStruct[5] Details

Name	Data Type	Offset	Notes
AlphaK1	fixed<0,10,0> (10 bits)	0	Constant Alpha value
reserved0	fixed<0,6,0> (6 bits)	10	Constant Alpha value
AlphaK2	fixed<0,10,0> (10 bits)	16	Constant Alpha value
reserved1	fixed<0,6,0> (6 bits)	26	
SrcFactCMatchSelect	fixed<0,3,0> (3 bits)	32	Blend Source Factor for Color if the color key comparison matches (BLEND_SRCFACTC)
reserved2	fixed<0,1,0> (1 bit)	35	
DstFactCMatchSelect	fixed<0,3,0> (3 bits)	36	Blend Destination Factor for Color if the color key comparison matches (BLEND_DSTFACTC)
reserved3	fixed<0,1,0> (1 bit)	39	
SrcFactAMatchSelect	fixed<0,3,0> (3 bits)	40	Blend Source Factor for Alpha if the color key comparison matches (BLEND_SRCFACTA)
reserved4	fixed<0,1,0> (1 bit)	43	
DstFactAMatchSelect	fixed<0,3,0> (3 bits)	44	Blend Source Factor for Alpha if the color key comparison matches (BLEND_DSTFACTA)
reserved5	fixed<0,1,0> (1 bit)	47	
reserved6	fixed<0,4,0> (4 bits)	48	
reserved7	fixed<0,4,0> (4 bits)	52	
reserved8	fixed<0,4,0> (4 bits)	56	
reserved9	fixed<0,4,0> (4 bits)	60	
reserved10	fixed<0,2,0> (2 bits)	64	
OverrideR	fixed<0,10,0> (10 bits)	66	Source RGBA override values

Name	Data Type	Offset	Notes
OverrideG	fixed<0,10,0> (10 bits)	76	
OverrideB	fixed<0,10,0> (10 bits)	86	
OverrideA	fixed<0,10,0> (10 bits)	96	
reserved11	fixed<0,2,0> (2 bits)	106	
UseOverrideR	fixed<0,1,0> (1 bit)	108	Enable source RGBA value override
UseOverrideG	fixed<0,1,0> (1 bit)	109	
UseOverrideB	fixed<0,1,0> (1 bit)	110	
UseOverrideA	fixed<0,1,0> (1 bit)	111	
MaskR	fixed<0,1,0> (1 bit)	112	Per-component blend output override enables (replace blend output with destination RGBA values)
MaskG	fixed<0,1,0> (1 bit)	113	
MaskB	fixed<0,1,0> (1 bit)	114	
MaskA	fixed<0,1,0> (1 bit)	115	
reserved12	fixed<0,12,0> (12 bits)	116	

13.4.3.10 FetchControl0Struct

Specifies various parameters related to scaling and filtering. Also specifies which input surfaces are enabled and the processing required for each slot.

Table 51: FetchControl0Struct Details

Name	Data Type	Offset	Notes
TargetRectLeft	fixed<0,14,0> (14 bits)	0	Target rectangle, restricts the output of pixels to this region
reserved0	fixed<0,2,0> (2 bits)	14	
TargetRectRight	fixed<0,14,0> (14 bits)	16	
reserved1	fixed<0,2,0> (2 bits)	30	
TargetRectTop	fixed<0,14,0> (14 bits)	32	
reserved2	fixed<0,2,0> (2 bits)	46	
TargetRectBottom	fixed<0,14,0> (14 bits)	48	
reserved3	fixed<0,2,0> (2 bits)	62	
Enable0	fixed<0,8,0> (8 bits)	64	Indicates which surfaces are enabled for fetching: Bit 0: Current field Bit 1: Previous field Bit 2: Next field Bit 3: Next field (noise filtered, takes priority over unfiltered field) Bit 4: Current motion field

Name	Data Type	Offset	Notes
			Bit 5: Previous motion field Bit 6: Previous previous motion field Bit 7: Combined motion field
Enable1	fixed<0,8,0> (8 bits)	72	
Enable2	fixed<0,8,0> (8 bits)	80	
Enable3	fixed<0,8,0> (8 bits)	88	
Enable4	fixed<0,8,0> (8 bits)	96	
DownsampleHoriz	fixed<0,11,0> (11 bits)	104	TargetWidth/DownsampleWidth (U9.2 used to lob bias filter). Set to 0 to enable filter override mode (mapping between stream and filter is explained in “Scaling and Filtering”)
reserved4	fixed<0,1,0> (1 bit)	115	
DownsampleVert	fixed<0,11,0> (11 bits)	116	TargetHeight/DownsampleHeight (U9.2 used to lob bias filter). Set to 0 to enable filter override mode (mapping between stream and filter is explained in “Scaling and Filtering”)
reserved5	fixed<0,1,0> (1 bit)	127	
FilterNoise0	fixed<0,10,0> (10 bits)	128	Strength of the spatial noise filter for slot 0. All detail and noise filter values (including chroma) become meaningless and should be set to 0 as soon as filter override is enabled
FilterDetail0	fixed<0,10,0> (10 bits)	138	Strength of the detail filter for slot 0
FilterNoise1	fixed<0,10,0> (10 bits)	148	Strength of the noise filter for slot 1
reserved6	fixed<0,2,0> (2 bits)	158	
FilterDetail1	fixed<0,10,0> (10 bits)	160	Strength of the detail filter for slot 1
FilterNoise2	fixed<0,10,0> (10 bits)	170	Strength of the noise filter for slot 2
FilterDetail2	fixed<0,10,0> (10 bits)	180	Strength of the detail filter for slot 2
reserved7	fixed<0,2,0> (2 bits)	190	
FilterNoise3	fixed<0,10,0> (10 bits)	192	Strength of the noise filter for slot 3
FilterDetail3	fixed<0,10,0> (10 bits)	202	Strength of the detail filter for slot 3
FilterNoise4	fixed<0,10,0> (10 bits)	212	Strength of the noise filter for slot 4
reserved8	fixed<0,2,0> (2 bits)	222	
FilterDetail4	fixed<0,10,0> (10 bits)	224	Strength of the detail filter for slot 4
reserved9	fixed<0,22,0> (22 bits)	234	
ChromaNoise0	fixed<0,10,0> (10 bits)	256	Strength of the noise filter for slot 0
ChromaDetail0	fixed<0,10,0> (10 bits)	266	Strength of the detail filter for slot 0
ChromaNoise1	fixed<0,10,0> (10 bits)	276	Strength of the noise filter for slot 1
reserved10	fixed<0,2,0> (2 bits)	286	

Name	Data Type	Offset	Notes
ChromaDetail1	fixed<0,10,0> (10 bits)	288	Strength of the detail filter for slot 1
ChromaNoise2	fixed<0,10,0> (10 bits)	298	Strength of the noise filter for slot 2
ChromaDetail2	fixed<0,10,0> (10 bits)	308	Strength of the detail filter for slot 2
reserved11	fixed<0,2,0> (2 bits)	318	
ChromaNoise3	fixed<0,10,0> (10 bits)	320	Strength of the noise filter for slot 3
ChromaDetail3	fixed<0,10,0> (10 bits)	330	Strength of the detail filter for slot 3
ChromaNoise4	fixed<0,10,0> (10 bits)	340	Strength of the noise filter for slot 4
reserved12	fixed<0,2,0> (2 bits)	350	
ChromaDetail4	fixed<0,10,0> (10 bits)	352	Strength of the detail filter for slot 4
reserved13	fixed<0,22,0> (22 bits)	362	
Mode0	fixed<0,4,0> (4 bits)	384	Deinterlace mode (DXVAHD_DEINTERLACE_MODE_PRIVATE)
AccumWeight0	fixed<0,3,0> (3 bits)	388	Accumulation weight for motion IIR filter for slot 0 (the default is 6). The first time an even/odd motion field is calculated, AccumWeight should be set to 0 to avoid having to clear the motion buffer beforehand.
Iir0	fixed<0,11,0> (11 bits)	391	Accumulation weight for noise reduction IIR filter for slot 0 (default value is 0x300).
reserved14	fixed<0,2,0> (2 bits)	402	
Mode1	fixed<0,4,0> (4 bits)	404	
AccumWeight1	fixed<0,3,0> (3 bits)	408	
Iir1	fixed<0,11,0> (11 bits)	411	
reserved15	fixed<0,2,0> (2 bits)	422	
Mode2	fixed<0,4,0> (4 bits)	424	
AccumWeight2	fixed<0,3,0> (3 bits)	428	
Iir2	fixed<0,11,0> (11 bits)	431	
reserved16	fixed<0,6,0> (6 bits)	442	
Mode3	fixed<0,4,0> (4 bits)	448	
AccumWeight3	fixed<0,3,0> (3 bits)	452	
Iir3	fixed<0,11,0> (11 bits)	455	
reserved17	fixed<0,2,0> (2 bits)	466	
Mode4	fixed<0,4,0> (4 bits)	468	
AccumWeight4	fixed<0,3,0> (3 bits)	472	

Name	Data Type	Offset	Notes
lir4	fixed<0,11,0> (11 bits)	475	
reserved18	fixed<0,8,0> (8 bits)	486	
OutputFlipX	fixed<0,1,0> (1 bit)	494	Flip output image horizontally
OutputFlipY	fixed<0,1,0> (1 bit)	495	Flip output image vertically
reserved19	fixed<0,10,0> (10 bits)	496	
reserved20	fixed<0,6,0> (6 bits)	506	

13.4.3.11 FetchControlCoeffStruct[520]

Specifies the coefficients used in the scalars for scaling/edge-enhancement/noise filtering. These coefficients are interpolated in the hardware to get the exact coefficients needed for the use case.

Table 52: FetchControlCoeffStruct[520] Details

Name	Data Type	Offset	
f00	fixed<0,10,0> (10 bits)	0	First coefficient of noise filter for 128-bit packet
f10	fixed<0,10,0> (10 bits)	10	First coefficient of normal scaling filter in 128-bit packet
f20	fixed<0,10,0> (10 bits)	20	First coefficient of detail filter in 128-bit packet
reserved0	fixed<0,2,0> (2 bits)	30	
f01	fixed<0,10,0> (10 bits)	32	Second coefficient of noise filter in 128-bit packet
f11	fixed<0,10,0> (10 bits)	42	Second coefficient of normal scaling filter in 128-bit packet
f21	fixed<0,10,0> (10 bits)	52	Second coefficient of detail filter in 128-bit packet
reserved1	fixed<0,2,0> (2 bits)	62	
f02	fixed<0,10,0> (10 bits)	64	Third coefficient of noise filter in 128-bit packet
f12	fixed<0,10,0> (10 bits)	74	Third coefficient of normal scaling filter in 128-bit packet
f22	fixed<0,10,0> (10 bits)	84	Third coefficient of detail filter in 128-bit packet
reserved2	fixed<0,2,0> (2 bits)	94	
f03	fixed<0,10,0> (10 bits)	96	Fourth coefficient of noise filter in 128-bit packet
f13	fixed<0,10,0> (10 bits)	106	Fourth coefficient of normal scaling filter in 128-bit packet
f23	fixed<0,10,0> (10 bits)	116	Fourth coefficient of detail filter in 128-bit packet
reserved3	fixed<0,2,0> (2 bits)	126	

13.4.4 Palette Structure

PaletteStruct sends palette information for indexed formats. Each slot has its own space in the palette buffer even if the format is not indexed. For non-indexed formats the space should left blank and should not be used by any other slots to send palette data. The number of palette entries for each slot can be at most 256.

13.4.5 History Buffer

The History buffer is used to store inter-frame communication data. It has two parts. The first part of the structure holds cadence-related information, and the second part stores histogram data for each slot.

The first Dword of the hist_control (control_vector) should be set by the driver. It informs Falcon whether to calculate the cadence.

13.4.6 FCE Microcode Buffer

This buffer holds the Fetch Control Engine (FCE) microcode that Falcon should load to the FCE unit. The driver should also pass the size of the FCE microcode with the SetFceUcodeSize() method.

13.4.7 Input Picture Buffers

Each picture is a separate buffer with its own surface offset method. The driver needs to send these offset methods for only the used pictures (source and reference) for the current execute.

13.4.8 Noise Reduced Picture Buffers

This buffer is used by the app to keep a noise-reduced picture data of future fields if DeNoise is enabled. In the next execute, noise-reduced picture data is used as the current field, and noise-reduced picture data of the previous execute will be used as previous picture. Noise reduction requires a forward and a backward reference field for interlaced content and a backward reference for progressive content.

In the NV24 case, the VIC app uses an additional surface to keep noise reduced field of the current picture. In this case, the current noise reduced surfaces are used as previous reference surfaces.

Noise-reduced picture buffer dimensions for a slot should be same as input buffer of that slot.

13.4.9 Motion Map Buffers

These buffers keep motion map data of the current and previous fields if MotionMap is enabled. The previous motion map buffer should be initialized to 255 (both luma and chroma) at the start of the clip. Use the motion map buffer of the previous execute pass prev motion map in the current execute for subsequent frames. MotionMap calculation is required for DiSi1 and also needs a forward and a backward reference. Behavior is not defined for progressive streams.

Motion map buffer dimensions for a slot should be same as input buffer of that slot.

13.4.10 Output Buffer

This buffer stores output data.

13.4.11 Method Naming and Programming

The VIC uses generic names to avoid confusing name overloading to the 16 surfaces (8 luma, 8 chroma) methods for the MAX_SLOTS

- SetSurfacexSlotyLumaOffset()
- SetSurfacexSlotyChromaOffset()

with x in [0..7] and y in [0..MAX_SLOTS]

The following sets of methods are required for a given ConfigStruct setup. Any setup not covered in this section should be regarded as illegal.

Note:

- NV24 frames are treated as NV12 fields. The frame format for NV24 has to be set to FRAME_FORMAT_TOP_FIELD.
- In case of noise reduction all previously noise reduced surfaces should be used as references instead of the original ones. In case of interlaced fields this also applies to the current fields as they have previously been noise reduced.
- Surface6 has been obsoleted and should not be used.
- History buffer is always required and should be owned by Falcon.

13.4.11.1 Progressive Frames

No Preprocessing

Enable bits have to be set to 0x01 in fetchControl0Struct.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()

Noise Reduction

Enable bits have to be set to 0x07 in fetchControl0Struct.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()
- SetSurface1SlotLumaOffset() (backward reference surface)
- SetSurface2SlotLumaOffset() (noise reduced surface)

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()
- SetSurface1SlotChromaOffset() (backward reference surface)
- SetSurface2SlotChromaOffset() (noise reduced surface)

13.4.11.2 Progressive Fields

No Preprocessing

Enable bits have to be set to 0x03 in fetchControl0Struct.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset() (top field)
- SetSurface1SlotLumaOffset() (bottom field)

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset() (top field)
- SetSurface1SlotChromaOffset() (bottom field)

Noise Reduction

Enable bits have to be set to 0x03f in fetchControl0Struct.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset() (top field)
- SetSurface1SlotLumaOffset() (bottom field)
- SetSurface2SlotLumaOffset() (top field of backward reference surface)
- SetSurface3SlotLumaOffset() (bottom field of backward reference surface)
- SetSurface4SlotLumaOffset() (top field of noise reduced surface)
- SetSurface5SlotLumaOffset() (bottom field of noise reduced surface)

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset() (top field)
- SetSurface1SlotChromaOffset() (bottom field)
- SetSurface2SlotChromaOffset() (top field of backward reference surface)
- SetSurface3SlotChromaOffset() (bottom field of backward reference surface)
- SetSurface4SlotChromaOffset() (top field of noise reduced surface)
- SetSurface5SlotChromaOffset() (bottom field of noise reduced surface)

13.4.11.3 Interlaced Frames

No Preprocessing

Enable bits have to be set to 0x01 in fetchControl0Struct.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()

Noise Reduction

Enable bits have to be set to 0x07 in fetchControl0Struct.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()
- SetSurface1SlotLumaOffset() (backward reference surface)
- SetSurface2SlotLumaOffset() (noise reduced surface)

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()
- SetSurface1SlotChromaOffset() (backward reference surface)
- SetSurface2SlotChromaOffset() (noise reduced surface)

13.4.11.4 Interlaced Fields

No Preprocessing

This is for BOB_FIELD only. For WEAVE, see “No Preprocessing” under “Progressive Fields“.

Enable bits have to be set to 0x01 in fetchControl0Struct.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()

Cadence Detection

Enable bits have to be set to 0x07 in fetchControl0Struct. This also requires cadence detection to be enabled.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()
- SetSurface1SlotLumaOffset() (backward reference field)
- SetSurface2SlotLumaOffset() (forward reference field)

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()
- SetSurface1SlotChromaOffset() (backward reference field)
- SetSurface2SlotChromaOffset() (forward reference field)

Motion Calculation

Enable bits have to be set to 0x17 in fetchControl0Struct. The data in the current motion field is read first and IIR filtered with the newly calculated motion before it is written back.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()
- SetSurface1SlotLumaOffset() (backward reference field)
- SetSurface2SlotLumaOffset() (forward reference field)
- SetSurface4SlotLumaOffset() (current motion field)

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()
- SetSurface1SlotChromaOffset() (backward reference field)
- SetSurface2SlotChromaOffset() (forward reference field)
- SetSurface4SlotChromaOffset() (current motion field)

Motion Calculation and Cadence Detection

Same as above in Motion Calculation except that cadence detection is enabled.

DISI1 (Motion Calculation and Motion Combine)

Enable bits have to be set to 0xb7 in fetchControl0Struct. The data in the current motion field is read first and IIR filtered with the newly calculated motion before it is written back. The combined motion field is generated out of current motion and previous motion to be used in DiSi1 deinterlacer.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()
- SetSurface1SlotLumaOffset() (backward reference field)
- SetSurface2SlotLumaOffset() (forward reference field)
- SetSurface4SlotLumaOffset() (current motion field)
- SetSurface5SlotLumaOffset() (previous motion field)
- SetSurface7SlotLumaOffset() (combined motion field)

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()
- SetSurface1SlotChromaOffset() (backward reference field)
- SetSurface2SlotChromaOffset() (forward reference field)
- SetSurface4SlotChromaOffset() (current motion field)
- SetSurface5SlotChromaOffset() (previous motion field)
- SetSurface7SlotChromaOffset() (combined motion field)

DISI1 (Motion Calculation and Motion Combine) and Cadence Detection

Same as above in DiSi1 except that cadence detection is enabled.

Noise Reduction

Enable bits have to be set to 0x0f in fetchControl0Struct.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()
- SetSurface1SlotLumaOffset() (previously noise reduced backward reference field)
- SetSurface2SlotLumaOffset() (forward reference field)
- SetSurface3SlotLumaOffset() (noise reduced forward reference field)

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()
- SetSurface1SlotChromaOffset() (previously noise reduced backward reference field)
- SetSurface2SlotChromaOffset() (forward reference field)
- SetSurface3SlotChromaOffset() (noise reduced forward reference field)

Noise Reduction and Cadence Detection

Same as above in Noise Reduction except that cadence detection is enabled.

Noise Reduction and Motion Calculation

Enable bits have to be set to 0x1f in fetchControl0Struct. The data in the current motion field is read first and IIR filtered with the newly calculated motion before it is written back.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()
- SetSurface1SlotLumaOffset() (backward reference field)
- SetSurface2SlotLumaOffset() (forward reference field)
- SetSurface3SlotLumaOffset() (noise reduced forward reference field)
- SetSurface4SlotLumaOffset() (current motion field)

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()
- SetSurface1SlotChromaOffset() (backward reference field)
- SetSurface2SlotChromaOffset() (forward reference field)
- SetSurface3SlotChromaOffset() (noise reduced forward reference field)
- SetSurface4SlotChromaOffset() (current motion field)

Noise Reduction and Motion Calculation and Cadence Detection

Same as above in Noise Reduction and Motion Calculation except that cadence detection is enabled.

Noise Reduction and DISI1 (Motion Calculation and Motion Combine)

Enable bits have to be set to 0xbf in fetchControl0Struct. The data in the current motion field is read first and IIR filtered with the newly calculated motion before it is written back. The combined motion field is generated out of current motion and previous motion to be used in DiSi1 deinterlacer.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()
- SetSurface1SlotLumaOffset() (backward reference field)
- SetSurface2SlotLumaOffset() (forward reference field)
- SetSurface3SlotLumaOffset() (noise reduced forward reference field)
- SetSurface4SlotLumaOffset() (current motion field)
- SetSurface5SlotLumaOffset() (previous motion field)
- SetSurface7SlotLumaOffset() (combined motion field)

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()
- SetSurface1SlotChromaOffset() (backward reference field)
- SetSurface2SlotChromaOffset() (forward reference field)
- SetSurface3SlotChromaOffset() (noise reduced forward reference field)
- SetSurface4SlotChromaOffset() (current motion field)
- SetSurface5SlotChromaOffset() (previous motion field)
- SetSurface7SlotChromaOffset() (combined motion field)

Noise Reduction and DISI1 (Motion Calculation and Motion Combine) and Cadence Detection

Same as above in Noise Reduction and DiSi1 except that cadence detection is enabled.

13.4.12 Application ID

Application ID is not needed as there is only one application for the engine (unlike different codecs for the MSDEC). But to use it with MSDEC infrastructure, the driver needs to pass the ApplicationId as 1 (corresponding to overlay 1).

13.4.13 Application-Specific Error Codes

The following table defines the error codes.

Table 53: Error Codes

Error Code	Value	Description
MSDECOS_ERROR_NONE	0x00000000	Default return code for app
MSDECOS_ERROR_EXECUTE_INSUFFICIENT_DATA	0x00000001	To be returned by the app to the OS
MSDECOS_ERROR_SEMAPHORE_INSUFFICIENT_DATA	0x00000002	Insufficient semaphore methods received. Method Data written out into MAILBOX1. MAILBOX0 bits 31:20- Method ID. MAILBOX0 bits 19-12: Method Count (SWFIFO + MTHDCOUNT register)
MSDECOS_ERROR_INVALID_METHOD	0x00000003	Unsupported method
MSDECOS_ERROR_INVALID_DMA_PAGE	0x00000004	Unsupported method
MSDECOS_ERROR_UNHANDLED_INTERRUPT	0x00000005	The app has either no interrupt handler or an unhandled OS error
MSDECOS_ERROR_EXCEPTION	0x00000006	Exception raised by Falcon
MSDECOS_ERROR_INVALID_CTXSW_REQUEST	0x00000007	Invalid CTXSW request to the OS
MSDECOS_ERROR_APPLICATION	0x00000008	Application returned a nonzero error code
MSDECOS_ERROR_SWBREAKPT	0x00000009	Exception raised to dump registers in debug mode

13.4.14 Application Method Registers

The VIC application method registers are accessed directly by the driver. Because the VIC uses the MSDEC infrastructure, two ranges of methods are used (in the MSDEC, there are common and application specific ranges). The range of method registers is:

- 0x700-0x73F (common method range) and
- 0x400-0x53F (method range corresponding to overlay 1)

The following table defines the method map.

Address	Method
0x0100	VIC.Nop
0x0140	VIC.PmTrigger
0x0200	VIC.SetApplicationID
0x0204	VIC.SetWatchdogTimer
0x0240	VIC.SemaphoreA
0x0244	VIC.SemaphoreB
0x0248	VIC.SemaphoreC

Address	Method
0x024c	VIC.CtxSaveArea
0x0250	VIC.CtxSwitch
0x0300	VIC.Execute
0x0304	VIC.SemaphoreD
0x0400	VIC.SetSurface0Slot0LumaOffset
0x0404	VIC.SetSurface0Slot0ChromaU_Offset
0x0408	VIC.SetSurface0Slot0ChromaV_Offset
0x040c	VIC.SetSurface1Slot0LumaOffset
0x0410	VIC.SetSurface1Slot0ChromaU_Offset
0x0414	VIC.SetSurface1Slot0ChromaV_Offset
0x0418	VIC.SetSurface2Slot0LumaOffset
0x041c	VIC.SetSurface2Slot0ChromaU_Offset
0x0420	VIC.SetSurface2Slot0ChromaV_Offset
0x0424	VIC.SetSurface3Slot0LumaOffset
0x0428	VIC.SetSurface3Slot0ChromaU_Offset
0x042c	VIC.SetSurface3Slot0ChromaV_Offset
0x0430	VIC.SetSurface4Slot0LumaOffset
0x0434	VIC.SetSurface4Slot0ChromaU_Offset
0x0438	VIC.SetSurface4Slot0ChromaV_Offset
0x043c	VIC.SetSurface5Slot0LumaOffset
0x0440	VIC.SetSurface5Slot0ChromaU_Offset
0x0444	VIC.SetSurface5Slot0ChromaV_Offset
0x0448	VIC.SetSurface6Slot0LumaOffset
0x044c	VIC.SetSurface6Slot0ChromaU_Offset
0x0450	VIC.SetSurface6Slot0ChromaV_Offset
0x0454	VIC.SetSurface7Slot0LumaOffset
0x0458	VIC.SetSurface7Slot0ChromaU_Offset
0x045c	VIC.SetSurface7Slot0ChromaV_Offset
0x0460	VIC.SetSurface0Slot1LumaOffset
0x0464	VIC.SetSurface0Slot1ChromaU_Offset
0x0468	VIC.SetSurface0Slot1ChromaV_Offset
0x046c	VIC.SetSurface1Slot1LumaOffset
0x0470	VIC.SetSurface1Slot1ChromaU_Offset
0x0474	VIC.SetSurface1Slot1ChromaV_Offset
0x0478	VIC.SetSurface2Slot1LumaOffset
0x047c	VIC.SetSurface2Slot1ChromaU_Offset
0x0480	VIC.SetSurface2Slot1ChromaV_Offset
0x0484	VIC.SetSurface3Slot1LumaOffset

Address	Method
0x0488	VIC.SetSurface3Slot1ChromaU_Offset
0x048c	VIC.SetSurface3Slot1ChromaV_Offset
0x0490	VIC.SetSurface4Slot1LumaOffset
0x0494	VIC.SetSurface4Slot1ChromaU_Offset
0x0498	VIC.SetSurface4Slot1ChromaV_Offset
0x049c	VIC.SetSurface5Slot1LumaOffset
0x04a0	VIC.SetSurface5Slot1ChromaU_Offset
0x04a4	VIC.SetSurface5Slot1ChromaV_Offset
0x04a8	VIC.SetSurface6Slot1LumaOffset
0x04ac	VIC.SetSurface6Slot1ChromaU_Offset
0x04b0	VIC.SetSurface6Slot1ChromaV_Offset
0x04b4	VIC.SetSurface7Slot1LumaOffset
0x04b8	VIC.SetSurface7Slot1ChromaU_Offset
0x04bc	VIC.SetSurface7Slot1ChromaV_Offset
0x04c0	VIC.SetSurface0Slot2LumaOffset
0x04c4	VIC.SetSurface0Slot2ChromaU_Offset
0x04c8	VIC.SetSurface0Slot2ChromaV_Offset
0x04cc	VIC.SetSurface1Slot2LumaOffset
0x04d0	VIC.SetSurface1Slot2ChromaU_Offset
0x04d4	VIC.SetSurface1Slot2ChromaV_Offset
0x04d8	VIC.SetSurface2Slot2LumaOffset
0x04dc	VIC.SetSurface2Slot2ChromaU_Offset
0x04e0	VIC.SetSurface2Slot2ChromaV_Offset
0x04e4	VIC.SetSurface3Slot2LumaOffset
0x04e8	VIC.SetSurface3Slot2ChromaU_Offset
0x04ec	VIC.SetSurface3Slot2ChromaV_Offset
0x04f0	VIC.SetSurface4Slot2LumaOffset
0x04f4	VIC.SetSurface4Slot2ChromaU_Offset
0x04f8	VIC.SetSurface4Slot2ChromaV_Offset
0x04fc	VIC.SetSurface5Slot2LumaOffset
0x0500	VIC.SetSurface5Slot2ChromaU_Offset
0x0504	VIC.SetSurface5Slot2ChromaV_Offset
0x0508	VIC.SetSurface6Slot2LumaOffset
0x050c	VIC.SetSurface6Slot2ChromaU_Offset
0x0510	VIC.SetSurface6Slot2ChromaV_Offset
0x0514	VIC.SetSurface7Slot2LumaOffset
0x0518	VIC.SetSurface7Slot2ChromaU_Offset
0x051c	VIC.SetSurface7Slot2ChromaV_Offset

Address	Method
0x0520	VIC.SetSurface0Slot3LumaOffset
0x0524	VIC.SetSurface0Slot3ChromaU_Offset
0x0528	VIC.SetSurface0Slot3ChromaV_Offset
0x052c	VIC.SetSurface1Slot3LumaOffset
0x0530	VIC.SetSurface1Slot3ChromaU_Offset
0x0534	VIC.SetSurface1Slot3ChromaV_Offset
0x0538	VIC.SetSurface2Slot3LumaOffset
0x053c	VIC.SetSurface2Slot3ChromaU_Offset
0x0540	VIC.SetSurface2Slot3ChromaV_Offset
0x0544	VIC.SetSurface3Slot3LumaOffset
0x0548	VIC.SetSurface3Slot3ChromaU_Offset
0x054c	VIC.SetSurface3Slot3ChromaV_Offset
0x0550	VIC.SetSurface4Slot3LumaOffset
0x0554	VIC.SetSurface4Slot3ChromaU_Offset
0x0558	VIC.SetSurface4Slot3ChromaV_Offset
0x055c	VIC.SetSurface5Slot3LumaOffset
0x0560	VIC.SetSurface5Slot3ChromaU_Offset
0x0564	VIC.SetSurface5Slot3ChromaV_Offset
0x0568	VIC.SetSurface6Slot3LumaOffset
0x056c	VIC.SetSurface6Slot3ChromaU_Offset
0x0570	VIC.SetSurface6Slot3ChromaV_Offset
0x0574	VIC.SetSurface7Slot3LumaOffset
0x0578	VIC.SetSurface7Slot3ChromaU_Offset
0x057c	VIC.SetSurface7Slot3ChromaV_Offset
0x0580	VIC.SetSurface0Slot4LumaOffset
0x0584	VIC.SetSurface0Slot4ChromaU_Offset
0x0588	VIC.SetSurface0Slot4ChromaV_Offset
0x058c	VIC.SetSurface1Slot4LumaOffset
0x0590	VIC.SetSurface1Slot4ChromaU_Offset
0x0594	VIC.SetSurface1Slot4ChromaV_Offset
0x0598	VIC.SetSurface2Slot4LumaOffset
0x059c	VIC.SetSurface2Slot4ChromaU_Offset
0x05a0	VIC.SetSurface2Slot4ChromaV_Offset
0x05a4	VIC.SetSurface3Slot4LumaOffset
0x05a8	VIC.SetSurface3Slot4ChromaU_Offset
0x05ac	VIC.SetSurface3Slot4ChromaV_Offset
0x05b0	VIC.SetSurface4Slot4LumaOffset
0x05b4	VIC.SetSurface4Slot4ChromaU_Offset

Address	Method
0x05b8	VIC.SetSurface4Slot4ChromaV_Offset
0x05bc	VIC.SetSurface5Slot4LumaOffset
0x05c0	VIC.SetSurface5Slot4ChromaU_Offset
0x05c4	VIC.SetSurface5Slot4ChromaV_Offset
0x05c8	VIC.SetSurface6Slot4LumaOffset
0x05cc	VIC.SetSurface6Slot4ChromaU_Offset
0x05d0	VIC.SetSurface6Slot4ChromaV_Offset
0x05d4	VIC.SetSurface7Slot4LumaOffset
0x05d8	VIC.SetSurface7Slot4ChromaU_Offset
0x05dc	VIC.SetSurface7Slot4ChromaV_Offset
0x0700	VIC.SetControlParams
0x0704	VIC.SetContextId
0x0708	VIC.SetContextIdForSlot0
0x070c	VIC.SetContextIdForSlot1
0x0710	VIC.SetContextIdForSlot2
0x0714	VIC.SetContextIdForSlot3
0x0718	VIC.SetContextIdForSlot4
0x071c	VIC.SetFceUcodeSize
0x0720	VIC.SetConfigStructOffset
0x0724	VIC.SetPaletteOffset
0x0728	VIC.SetHistOffset
0x072c	VIC.SetFceUcodeOffset
0x0730	VIC.SetOutputSurfaceLumaOffset
0x0734	VIC.SetOutputSurfaceChromaU_Offset
0x0738	VIC.SetOutputSurfaceChromaV_Offset
0x073c	VIC.SetPictureIndex
0x1114	VIC.PmTriggerEnd

13.5 VIC THI Registers

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

13.5.1.1 NV_PVIC_THI_INCR_SYNCPT

Offset: 0x0 │ Read/Write: R/W │ Reset: 0x00000000

Bit	R/W	Reset	Description
31:16	R	0	Reserved
15:8	R/W	0	NV_PVIC_THI_INCR_SYNCPT_COND: 0: COND_IMMEDIATE (default) 1: COND_OP_DONE

Bit	R/W	Reset	Description
7:0	R/W	0	NV_PVIC_THI_INCR_SYNCPT_INDX: 0: INDX_INIT (default)

13.5.1.2 NV_PVIC_THI_INCR_SYNCPT_ERR

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

Bit	R/W	Reset	Description
31:4	R	0	Reserved
1	RW1C	0	NV_PVIC_THI_INCR_SYNCPT_ERR_COND_STS_OPDONE: 0: COND_STS_OPDONE_INIT (default) 1: COND_STS_OPDONE_CLEAR
0	RW1C	0	NV_PVIC_THI_INCR_SYNCPT_ERR_COND_STS_IMM: 0: COND_STS_IMM_INIT (default) 1: COND_STS_IMM_CLEAR

13.5.1.3 NV_PVIC_THI_CTXSW_INCR_SYNCPT

Offset: 0xc | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:8	R	0	Reserved
7:0	R/W	0	NV_PVIC_THI_CTXSW_INCR_SYNCPT_INDX: 0: INDX_INIT (default)

13.5.1.4 NV_PVIC_THI_CTXSW

Offset: 0x20 | Read/Write: R/W | Reset: 0x0000F800

Bit	R/W	Reset	Description
31:28	R/W	0	NV_PVIC_THI_CTXSW_NEXT_CHANNEL: 0: NEXT_CHANNEL_INIT (default)
27:26	R	0	Reserved
25:16	R/W	0	NV_PVIC_THI_CTXSW_NEXT_CLASS: 0: NEXT_CLASS_INIT (default)
15:12	R/W	Fh	NV_PVIC_THI_CTXSW_CURR_CHANNEL: Fh CURR_CHANNEL_INIT (default)
11	R	1	NV_PVIC_THI_CTXSW_AUTO_ACK: 1: AUTO_ACK_INIT (default)
10	R	0	Reserved
9:0	R/W	0	NV_PVIC_THI_CTXSW_CURR_CLASS: 0: CURR_CLASS_INIT (default)

13.5.1.5 NV_PVIC_THI_CONT_SYNCPT_EOF

Offset: 0x28 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:9	R	0	Reserved
8	R/W	0	NV_PVIC_THI_CONT_SYNCPT_EOF_COND: 0: COND_INIT (default)

Bit	R/W	Reset	Description
7:0	R/W	0	NV_PVIC_THI_CONT_SYNCPT_EOF_INDEX: 0: INDEX_INIT (default)

13.5.1.6 NV_PVIC_THI_METHOD0

Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000

There are two method registers METHOD0 and METHOD1

Bit	R/W	Reset	Description
31:12	R	0	Reserved
11:0	R/W	0	NV_PVIC_THI_METHOD0_OFFSET: 0X 0: OFFSET_INIT (default)

13.5.1.7 NV_PVIC_THI_METHOD1

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

Bit	R/W	Reset	Description
31:0	R/W	0	NV_PVIC_THI_METHOD1_DATA: 0: DATA_INIT (default)

13.5.1.8 NV_PVIC_THI_INT_STATUS

Offset: 0x78 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:1	R	0	Reserved
0	RW1C	0	NV_PVIC_THI_INT_STATUS_FALCON_INT: 0: FALCON_INT_INIT (default) 1: FALCON_INT_CLEAR

13.5.1.9 NV_PVIC_THI_INT_MASK

Offset: 0x7c | Read/Write: R/W | Reset: 0x00000001

Bit	R/W	Reset	Description
31:1	R	0	Reserved
0	R/W	1	NV_PVIC_THI_INT_MASK_FALCON_INT: 1: FALCON_INT_INIT (default)

13.6 HOSTIF Miscellaneous Registers

13.6.1 NV_PVIC_FALCON_ITFEN

Offset: 0x1048 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:3	R	0	Reserved
2	R/W	0	NV_PVIC_FALCON_ITFEN_PRIV_POSTWR. Indicates whether to use a post write on the main priv interface. By default, all writes to the Control and Status Bus (CSB) from the main priv interface are non-posted to support error reporting. 0: PRIV_POSTWR_INIT (default)

Bit	R/W	Reset	Description
			0: PRIV_POSTWR_FALSE 1: PRIV_POSTWR_TRUE
1	R/W	0	NV_PVIC_FALCON_ITFEN_MTHDEN. Method interface enable. When set, allows the host to push methods into the method FIFO 0: MTHDEN_INIT (default) 0: MTHDEN_DISABLE 1: MTHDEN_ENABLE
0	R/W	0	NV_PVIC_FALCON_ITFEN_CTXEN. Context switch interface enable. When set, allows the context switch state machine to react to incoming context switch requests from the host. 0: CTXEN_INIT (default) 0: CTXEN_DISABLE 1: CTXEN_ENABLE

13.7 Falcon UCTL Registers

13.7.1 NV_PVIC_FALCON_CPUCTL

Offset: 0x1100 │ Read/Write: R/W │ Reset: 0x000000XX

Bit	R/W	Reset	Description
31:6	R	0	Reserved
5	R	Unknown	NV_PVIC_FALCON_CPUCTL_STOPPED: Indicates whether the CPU is currently in the stopped state. Falcon exits this state if a 1 is written to the STARTCPU bit or if an interrupt arrives on one of its 2 inputs and the corresponding IE bit in CSW is set. 1: STOPPED_TRUE 0: STOPPED_FALSE
4	R	Unknown	NV_PVIC_FALCON_CPUCTL_HALTED: Indicates whether the CPU is currently in the halted state. Falcon can only exit this state when a 1 is written to the STARTCPU bit. 1: HALTED_TRUE 0: HALTED_FALSE
3	RW1C	Unknown	NV_PVIC_FALCON_CPUCTL_HRESET: Set to TRUE to apply hard reset. This bit will auto-clear. Setting it FALSE has no effect. 1: HRESET_TRUE 0: HRESET_FALSE
2	RW1C	Unknown	NV_PVIC_FALCON_CPUCTL_SRESET: Set to TRUE to apply soft reset. This bit will auto-clear. Setting it FALSE has no effect. 1: SRESET_TRUE 0: SRESET_FALSE
1	RW1C	Unknown	NV_PVIC_FALCON_CPUCTL_STARTCPU: Set STARTCPU to TRUE to start CPU execution while in a HALTED state. If a start request is still pending, setting to FALSE will cancel the start request. Writing any value has no effect while the CPU is running. 1: STARTCPU_TRUE 0: STARTCPU_FALSE
0	RW1C	Unknown	NV_PVIC_FALCON_CPUCTL_IINVAL: Set to TRUE to mark all blocks in IMEM except block 0 as INVALID. This bit will auto-clear. Setting to FALSE has no effect. 1: IINVAL_TRUE 0: IINVAL_FALSE

13.7.2 NV_PVIC_FALCON_BOOTVEC

Offset: 0x1104 │ Read/Write: R/W │ Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	NV_PVIC_FALCON_BOOTVEC_VEC: Stores the initial execution start address of the CPU when it is first started after a reset.

Bit	R/W	Reset	Description
			0: VEC_INIT (default)

13.8 Falcon DMA Registers

13.8.1 NV_PVIC_FALCON_DMACTL

Offset: 0x110c | Read/Write: R/W | Reset: 0x000000XX

Bit	R/W	Reset	Description
31:7	R	0	Reserved
6:3	R	None	NV_PVIC_FALCON_DMACTL_DMAQ_NUM: Indicates valid request number at DMA request queue.
2:1	R	0	Reserved
0	R/W	1	NV_PVIC_FALCON_DMACTL_REQUIRE_CTX: When set to TRUE, a valid context must be loaded before any DMA request can be serviced. Pending requests without a valid current context remain pending, and do not prevent the engine from reporting idle. When clear, DMA requests are serviced regardless of the current context. Once a request is issued, it must complete before the engine can report idle, as needed for example to process WFI context switch requests. 1: REQUIRE_CTX_INIT (default) 1: REQUIRE_CTX_TRUE 0: REQUIRE_CTX_FALSE

13.8.2 NV_PVIC_FALCON_DMATRFBASE

Offset: 0x1110 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	NV_PVIC_FALCON_DMATRFBASE_BASE: 0: BASE_INIT (default)

13.8.3 NV_PVIC_FALCON_DMATRFMOFFS

IMEM/DMEM offset for the transfer.

Offset: 0x1114 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:16	R	0	Reserved
15:0	R/W	0	NV_PVIC_FALCON_DMATRFMOFFS_OFFS: 0: OFFS_INIT (default)

13.8.4 NV_PVIC_FALCON_DMATRFCMD

Offset: 0x1118 | Read/Write: R/W | Reset: 0x0000XXXX

Bit	R/W	Reset	Description
31:15	R	0	Reserved
14:12	R/W	Unknown	NV_PVIC_FALCON_DMATRFCMD_CTXDMA:
11	R	0	Reserved
10:8	R/W	Unknown	NV_PVIC_FALCON_DMATRFCMD_SIZE: 0: SIZE_4B

Bit	R/W	Reset	Description
			1: SIZE_8B 2: SIZE_16B 3: SIZE_32B 4: SIZE_64B 5: SIZE_128B 6: SIZE_256B
7:6	R	0	Reserved
5	R/W	Unknown	NV_PVIC_FALCON_DMATRFCMD_WRITE: 1: WRITE_TRUE 0: WRITE_FALSE
4	R/W	Unknown	NV_PVIC_FALCON_DMATRFCMD_IMEM: 1: IMEM_TRUE 0: IMEM_FALSE
3:2	R	0	Reserved
1	R	Unknown	NV_PVIC_FALCON_DMATRFCMD_IDLE: Indicates that the DMA engine is still busy with a transfer or has more transfers pending in the queue. 1: IDLE_TRUE 0: IDLE_FALSE
0	R	Unknown	NV_PVIC_FALCON_DMATRFCMD_FULL: Indicates that the DMA request queue is full and a valid request is still needed to move into the queue. 1: FULL_TRUE 0: FULL_FALSE

13.8.5 NV_PVIC_FALCON_DMATRFFBOFFS

Frame buffer (FB) offset for the transfer.

Offset: 0x111c │ Read/Write: R/W │ Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	NV_PVIC_FALCON_DMATRFFBOFFS_OFFS: 0: OFFS_INIT (default)

[THIS PAGE INTENTIONALLY LEFT BLANK]

14.0 CPU

The NVIDIA® Tegra® K1 processor CPU complex contains quad ARM® Cortex®-A15 CPUs in a 4-PLUS-1 configuration with a fifth architecturally identical power-saving Cortex-A15 Companion Core.

14.1 Cortex-A15 CPU

The Cortex-A15 CPU is an advanced processor design with many features for high instruction throughput. It integrates the L2 cache controller into the CPU complex unlike Cortex-A9 CPUs. All of the CPUs include the NEON Media Processing Engine. Further details of the Cortex-A15 CPU itself are available from ARM.

These two documents are the key Cortex-A15 references, and both are available from ARM's website:

- Cortex-A15
  Revision: r3p3
  Technical Reference Manual

  Published by ARM Limited, document number ARM DDI 0438D.
- ARM Architecture Reference Manual
  ARM v7-A and ARM v7-R edition

  Published by ARM Limited, document number ARM DDI 0406C.

> **Note:** These are the current versions at the time of writing. You should periodically look for updates and always refer to the latest available version of this material.

You should also review the latest version of the ARM errata for Cortex-A15 CPUs. ARM has published this on their website as the Software Developers' Errata Notice.

14.1.1 Cortex-A15 Implementation Details

Many aspects of the Cortex-A15 CPU may be configured for a specific implementation. The Tegra K1 implementation has the following features:

Cortex-A15 Processor Configuration		
	Option	
Feature	Main Quad-Core Complex	Companion Core
Processor Revision	r3p3	r3p3
Number of processors	4	1
L2 Cache Size	2MB	512KB
L2 Arbitration register slice	1	0
L2 Tag RAM register slice	1	1
L2 Data RAM register slice	1	1
L2 Tag RAM latency	2	2
L2 Data RAM latency	3 cycle	2 cycle
L2 Logic idle gated clock	Included	Included
ECC/parity support	None	None
VFP and Neon	Included	Included

Cortex-A15 Processor Configuration		
	Option	
Feature	Main Quad-Core Complex	Companion Core
Generic Interrupt Controller	Included	Included
Shared Peripheral Interrupts	160	160

14.2 4-Plus-1 Configuration and Control

The NVIDIA 4-PLUS-1 processor architecture has a fifth Companion Core available for low power operation. In addition to the main quad core CPU complex, the Companion Core is an additional CPU complex replicating CPU 0 of the main complex, and implemented in a low-leakage technology. The Companion Core is also sometimes referred to by its engineering name, the “shadow” core; and also is commonly referred to as the LP Core or LP CPU Cluster referring to its Low Power operating capability.

The goal of this architecture is to provide efficient CPU operation across a very broad range of processor loads, with the quad-core complex optimized for high-speed operation with low dynamic power, and the Companion Core optimized for low-speed operation where transistor leakage becomes an important factor in power. These two CPU complexes operate exclusively to each other, meaning the hardware does not support simultaneous operation in any form. The main quad core complex has 2 Megabytes of L2 cache RAM, the companion core as a separate 512 Kilobyte L2 cache RAM.

The quad core complex implements separate power-gating for each of the four cores, and also a fifth power gate domain for the shared logic and L2.

Refer to the Flow Controller section of this document for details on how this is controlled.

14.3 Exclusive Operations

The Tegra K1 processor supports ARM exclusive operations (LDREX*, STREX* and CLREX) in cacheable memory only. Device or Strongly-Ordered memory spaces should not be used with these instructions.

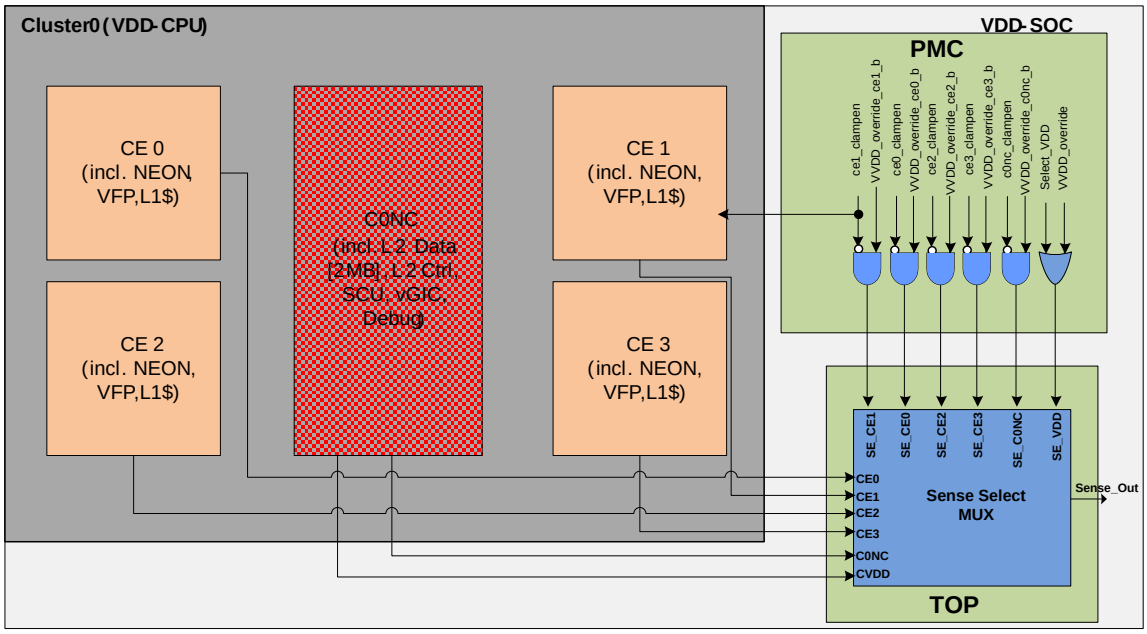
14.4 CPU Voltage Sensing Control

The Voltage Regulator (VR) delivers power to the board, then to the package and to the transistors in the silicon. The VR regulates its voltage output based on the sense feedback. To reduce inaccuracy in the VR output voltage level, the sense voltage needs to be as close to the transistors as possible.

For a power-gated partition, if the sense point is at the real VDD power grid, then the drop across the power-gate adds to the inaccuracy of the sense voltage. To reduce this inaccuracy, virtual VDD can be sensed. However, if the PG partition is power-gated, then its virtual VDD is off and cannot be used for sense feedback. Because partitions are power-gated/ungated dynamically, there needs to be control logic to provide dynamic control for the voltage-sense mux such that the virtual VDD is selected only when the corresponding PG partition is power-ungated.

As shown in the following diagram, a 6-input mux is used to select one (or more) of sensed voltages. The 6-input voltages are VVDD of each of the 5 PG partitions (in cluster0) and the real CPU VDD. The mux is expected to select the VVDD of all PG partitions that are power-ungated (powered on). If none of the 5 PG partitions are power-ungated, then real VDD is selected as the sense voltage.

Figure 26: CPU Voltage Sensing Mux Control Signals



The PMC provides a 6-bit mux select bus. The mux corresponding to 5 partitions is selected if those partitions are powered on. During normal operation, selection of the correct voltage sense signal is handled entirely by the PMC. For debug purposes, a per partition override register (see the APBDEV_PMC_CPU_VSENSE_OVERRIDE_0 register) is provided which can be configured to disable input from any of the 5 PG partitions, and select real VDD. The sensing mux select control needs to be as follows:

- Never select the VVDD of a partition which is power-gated
- Select VVDD of all partitions that are power-ungated
- When all partitions are power-gated, select real VDD
  - During transition, both power-ungated (VVDD) and real VDD can be selected.

[THIS PAGE INTENTIONALLY LEFT BLANK]

15.0 FLOW CONTROLLER

The flow controller provides the sequencing of hardware-controlled CPU power states for the main CPU complex and the AVP coprocessor (also known as COP). It also provides hardware support for CPU cluster switching between the fast quad-core complex and the companion core.

The flow controller receives CPU power-state requests from CPUs, prioritizes them, and sends power on/off requests to PMC, which powers gates/ungates corresponding CPUs. It monitors per CPU interrupts and events to determine CPU wake condition and initiates the CPU wake based on wake events/interrupts.

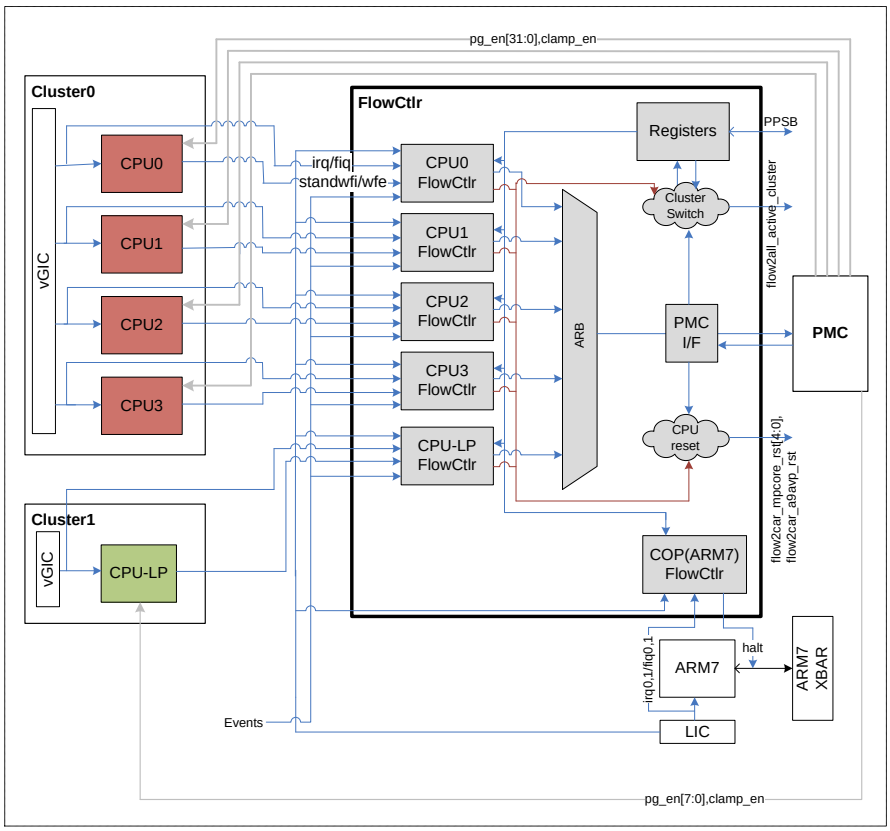
The flow controller provides configuration registers for software to configure wake events, etc. The flow controller provides a mechanism to halt conditionally or unconditionally:

- Conditional halt/resume is based on a variety of events, including timers, external trigger, and internal clocks.
- COP (AVP) is halted by extending a bus cycle.
- CPU can be halted by software interaction: software issues the WFI instruction to stop the processor, the flow controller asserts the EVENT input of the CPU to restart it.

15.1 Features and Functionality

The flow controller receives requests from processors (Cortex®-A15 processors and the AVP) to change their power state, or system power state. It also receives the interrupts from the interrupt controllers (vGIC and LIC) and other events which it uses to wake them up. It also provides support for cluster switching.

Figure 27: Flow Controller Block Diagram



15.1.1 WFI Instructions and STANDBYWFI Signals Usage

Tegra K1 devices use the WFI (Wait For Interrupt) instruction of ARMv7-A architecture (supported by both Cortex-A15 and Cortex-A9 processors), used by the CPU to interlock with the flow controller for the purpose of triggering low-power-state transitions.

When executing WFI, the processor waits for all prior instructions to complete before entering the idle (low-power) state. The WFI instruction ensures that all explicit memory accesses that are prior to the WFI instruction in program order have completed. Also, WFI ensures that all speculative memory accesses have completed.

After executing WFI, the processor asserts the STANDBYWFI signal. Assertion of STANDBYWFI guarantees that the processor is in idle, and there is no outstanding memory access. The processor continues to assert STANDBYWFI until one of the interrupt (IRQ or FIQ) is asserted or processor is reset.

The flow controller can be configured to trigger (per CPU) power-gating based on WFI instruction execution.

15.1.2 CPU (A15) Low-Power States

15.1.2.1 CPU Clock Gating

If a CPU is idle for a short period of time, then it can be halted which results in its CPU clock-gating. This is referred to as LP3 CPU power-state. The WFI instruction is executed for putting CPU into LP3 (i.e. for clock-gating). After executing WFI, CPU disables most of the clocks in the CPU except the clock for the small wake up logic clock.

Exit from halt state occurs when the CPU detects an interrupt on IRQ or FIQ pins. Note that this state is per CPU. This clock-gating is internal to the Cortex-A15 processor, so it does not require any support from the flow controller.

15.1.2.2 CPU Power Gating

The flow controller uses a (separate) state machine to sequence power-gating for each CPU. The main CPUs flow-controller state machines are different from that for the CPU Rail PwrUp and COP. So it has three different variants of state machines: one for main CPUs (which is instantiated 4 times, one for each CPU), one for CPU Rail PwrUp and one for COP. If the active cluster is cluster1, then based on CPU-ID, corresponding state machine is used for CPU-LP.

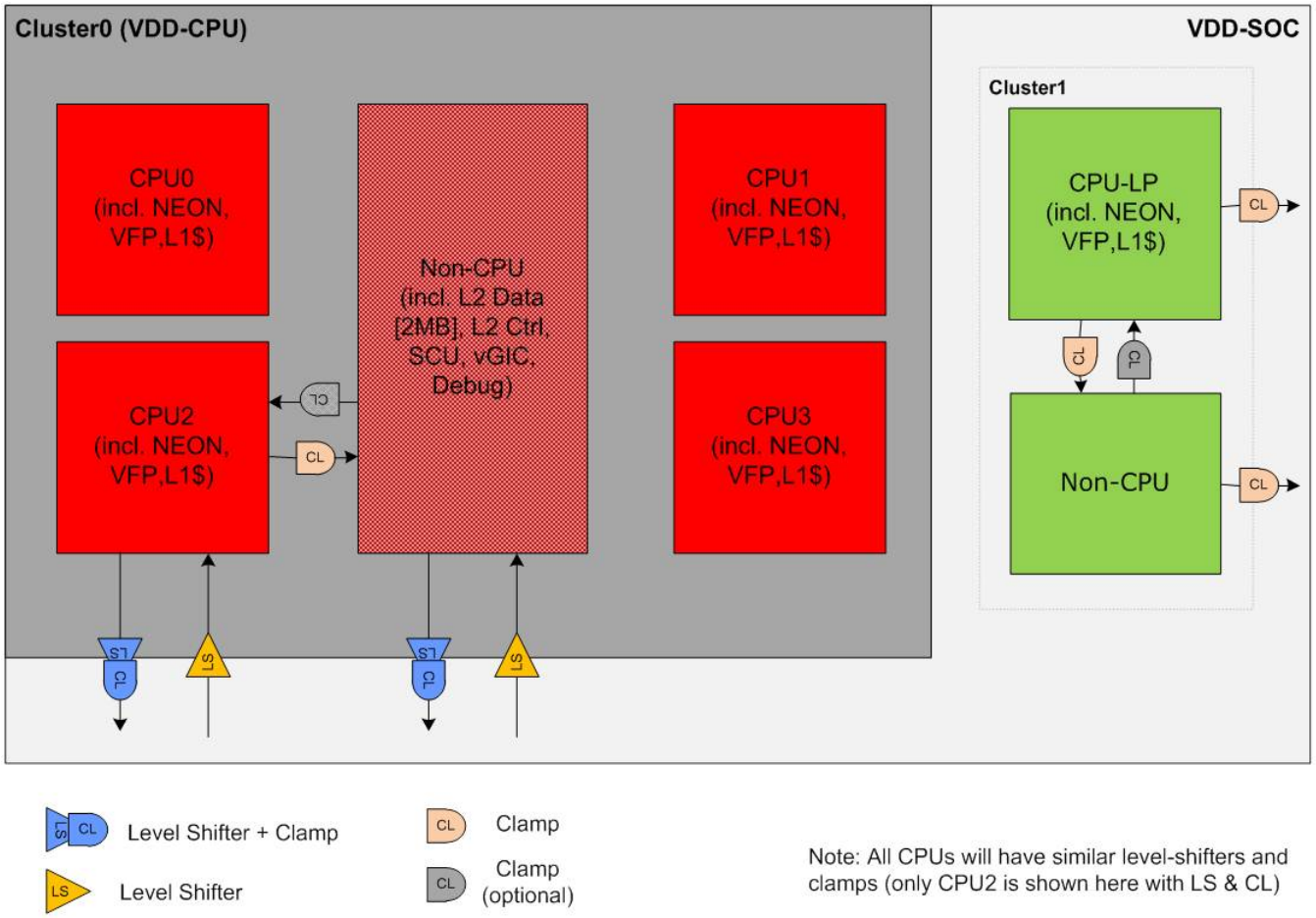
The flow controller can have one or more power-gating requests outstanding. However, it only issues one request at a time to the PMC. A fixed priority arbiter is used to arbitrate among CPU power-gating on/off requests, and the winning request is sent to the PMC. This is primarily for two reasons: i) to avoid power noise issues, only one partition is power gated/ungated at a time, ii) PMC's power-gating controller supports only one request at a time.

The flow controller to PMC interface includes core-ID which explicitly indicates which CPU or non-CPU partition should be power gated/ungated. PMC is not expected to use cluster-ID for identifying whether request is for cluster0 or cluster1 CPU.

CPU Power Gated Domains

As shown in the figure below, each CPU is an independent power-gated domain. The cluster0 non-CPU may be power gated as a separate domain. The cluster1 non-CPU is power-gated as a separate domain.

Figure 28: CPUs PG Domains



15.1.2.3 Cluster0 CPU-Rail Gating

At boot, CPU rail is off by default, and is powered on by the AVP boot code using a direct register write to PMC (or by direct I2C commands to PMIC). Also, at LP0 exit, the CPU rail (if need be) is powered-on (by software) by direct register write to the PMC (or by direct I2C commands to PMIC).

The CPU rail can be powered off by software running on the companion core after a cluster switch using a direct register write to PMC. Refer to the “Cluster Switch” section for details.

15.1.2.4 Cluster0 CPU Power Gating

Each of the cluster0 CPUs can be independently power-gated with the help of the flow controller. If the CPU being power-gated is not the last one (or is not requesting the CPU rail to be powered off), then it is power gated/ungated as described in the section on “CPU Power Gating/Ungating Sequence”.

The last CPU being power-gated can request non-CPU to be power-gated or CPU rail to be powered off. Similar when the first CPU is being woken up, then the flow controller would power on (if it was off) the CPU rail or power-ungate non-CPU (if it was power-gated).

Note:	The flow controller will either sequence the CPU rail controls or the non-CPU power-gating controls (with last CPU power-gating request), but not both.

The flow controller can be set up to power gate/ungate each of the CPU independently. In addition, the flow controller can be set up to power gate/ungate non-CPU as a result of a CPU power gate/ungate request. The following section describes a non-

CPU power gate/ungate along with CPU. The individual CPU only power gate/ungate is described in the section on “CPU Power Gating/Ungating Sequence”.

The following table indicates power-on options which can be triggered when the flow controller receives CPU wakeup event.

Per CPU Power Ungate	Non-CPU Power UnGate	CPU Rail Power On	Comments
Yes	No	No	Individual CPU power-ungating
Yes	Yes	No	

15.1.2.5 Cluster1 CPU Power Gating

The cluster1 CPU (also known as shadow CPU or CPU-LP) can be power gated/ungated with the help of the flow controller. Along with CPU-LP power-gating, its non-CPU can be power gated (by configuring flow controller before requesting CPU power-gating). From PMC perspective, CPU-LP and non-CPU are two separate power-gated domains. The flow controller sends CPU-LP and non-CPU power-gating requests to the PMC as follows:

The flow controller can be set up to power gate/ungate CPU-LP only. In addition, the flow controller can be set up to power gate/ungate non-CPU along with CPU-LP as result of CPU-LP power gate/ungate request. The next section describes non-CPU power gate/ungate along with CPU-LP. The CPU-LP only power gate/ungate is described in the section on “CPU Power Gating/Ungating Sequence”.

Power-Gating (Power-Off) CPU and Non-CPU

To initiate CPU-LP power-gating, SOFTWARE configures the flow-controller registers associated with CPU-ID (in MPID register). The SOFTWARE initiates CPU-LP (with non-CPU) power-gating request as described in section on CPU Power-Gating (Power OFF) Sequence. After that, the flow controller sequences CPU-LP and non-CPU power-gatings as follows:

- [SW-CPU] Initiate CPU-LP power gating with non-CPU power-gating rail (refer to the section on CPU Power-Gating (Power OFF) Sequence) for details of software steps
- [HW-Flow] CPU power-gating step
  - Request PMC to power-gate the CPU (Refer to the section on “CPU Power-Gating (Power OFF) Sequence”)
  - Wait for PMC acknowledgment
- [HW-Flow] Non-CPU power-gating step
  - If non-CPU is off (see CPU_PWR_CSR[C1NC_STS]), or in the process of being powered on or off then this step is done.

    Note: This condition should never be seen. But this makes the cluster1 sequence the same as that of cluster0, so it is all right to make this check in the state machine.
  - Update non-CPU power status register (CPU_PWR_CSR[C1NC_STS]) to be in the process powering off
  - Request PMC to power-gate non-CPU (refer to the section on “Non-CPU Power Gating/Ungating Sequence”)
  - Waits for PMC acknowledgment
  - Update non-CPU power status (CPU_PWR_CSR[C1NC_STS]) to be powered off

Power-Ungating (Power On) CPU and Non-CPU

Based on CPU-LP wake-up event ((see HALT_CPUx_EVENT register), the flow controller power-ungates CPU-LP (and non-CPU).

- [HW-Flow] Receives CPU-LP wake event
- [HW-Flow] Non-CPU power-ungating step
  - If non-CPU is power-ungated, then this step is done. Else,

- If non-CPU is in the process of being powered on, then wait for it to be powered on. After that, this step is done. Else,
- If non-CPU is in the process of being powered off, then wait for it to be powered off. After that continue to power it off.

> **Note:** The above two steps should never be encountered. But making these checks is all right and makes this sequence similar to corresponding cluster0 sequence.

- Update non-CPU power status register (CPU_PWR_CSR[C0NC_STS]) to be in the process of powering on.
- Request PMC to power-ungate non-CPU
- Wait for PMC acknowledgment
- Update non-CPU power status register (CPU_PWR_CSR[RAIL_STS]) to be powered on.

- [HW-Flow] CPU power-ungating step
  - Request PMC to power-ungate CPU
  - Waits for PMC ack

15.1.3 Cluster Switch

15.1.3.1 Cluster1 (Shadow) to Cluster0 Switch

The cluster switch is a variant of LP2 low-power state transition. From usage perspective, cluster1 to cluster0 switch is latency critical. For cluster1 to cluster0 switch, the cluster1 CPU is power-gated (with its non-CPU power-gating), cluster switched is performed, and a cluster0 CPU is powered on (either immediately or conditional on a wake event).

Following is the cluster1 to cluster0 migration sequence:

- [SW-CPU] Configure the flow controller for cluster switch
- [SW-CPU] Configure cluster0 CPU wake event
  - Software has to configure wake event for the CPU which is in the CPU-ID. After cluster switch, the flow controller can only wake-up the CPU which is pointed to by CPU-ID.
- [SW-CPU] Trigger CPU rail on sequence as described in “CPU Rail Gating/Ungating Sequence”.

> **Note:** This step allows the CPU rail (cluster0 power) to be powered up in parallel with the cluster1 power-gate sequence. However, cluster1 is still an active cluster (until the cluster state is switched by the flow controller).

- [SW-CPU] Initiate power-gating of cluster1 CPU (with non-CPU)
- [HW-Flow] Perform cluster1 non-CPU and CPU-LP power-gating as described in the section on “Cluster1 CPU Power Gating”
- [HW-Flow] After cluster1 power-gating, wait for pre-cluster-switch delay (programmed in a flow-controller register)
- [HW-Flow] Perform cluster switch
- [HW-Flow] Wait for post-cluster-switch-delay (programmed in a flow-controller register)
- [HW-Flow] Waits for the wake event (this could be immediate)
- [HW-Flow] Power-on first CPU of cluster0 as described in the section on Cluster0 CPU Power Gating

15.1.3.2 Cluster0 to Cluster1 (Shadow) Switch

The cluster switch is a variant of LP2 low-power state transition. For cluster0 to cluster1 switch, the last CPU of cluster0 is power-gated, the cluster is switched, and cluster1 CPU is power ungated (either immediately or conditional on wake event).

Following is the cluster0 to cluster1 migration sequence:

- [SW-CPU] Configure the flow controller for cluster switch

- [SW-CPU] Configure cluster1 CPU wake event
- [SW-CPU] Configure the CPU-ID of CPU which is being migrated to CPU-LP
- [SW-CPU] Initiate power-gating of last CPU (with non-CPU power gatingoption) as described in section on “Cluster0 CPU Power Gating” for CPU power-gating and non-CPU power gating.
- [HW-Flow] Wait for pre-cluster-switch delay (programmed in a flow-controller register)
- [HW-Flow] Perform cluster switch
- [SW-CPU] Software running on cluster 1 can power down the CPU rail by directly writing to the CRAIL bit of the APBDEV_PMC_PWRGATE_TOGGLE_0 register. The CPU rail has to be brought up by software prior to switching the cluster back to cluster 0 using the same mechanism.
- [HW-Flow] Wait for post-cluster-switch-delay (programmed in a flow-controller register)
- [HW-Flow] Waits for the wake event (if needed)
  - At the wake event, the flow controller powers up cluster1 as described in section on “Cluster1 CPU Power Gating”

15.1.4 CPU-ID Function

When the active cluster is cluster1, then the flow controller uses CPU-ID (programmed in MPID register) to associate CPU-LP to corresponding CPU state machine which is used to power gate/ungate CPU-LP. Also, during cluster1 to cluster0 switch, flow-controller uses CPU-ID to wake the corresponding cluster0 CPU (based on configured wake events).

Also, CPU-ID is used to drive cluster1 CPU-ID pins, which are captured in the CPU (CP15) register that software uses to read the CPU-ID of the current CPU. Note that the CP15 register captures the MPID value, which is the virtualized MPIDR, not the physical MPIDR. Thus, it must be read by the CPU in monitor mode or virtualization mode.

Once cluster0 is active and cluster migration has been completed, then the flow controller does not have any use for CPU-ID.

15.1.5 AVP Low-Power States

15.1.5.1 Halt (Clock Gating)

There is no specific halt instruction in the AVP. However, its memory bus cycle can be extended by any number of cycles.

When configured to put the AVP into halt, the flow controller forces its memory bus interface into wait-state. After that if the AVP issues any bus cycle (e.g., fetching next instruction), then that bus cycle does not return unless flow-controller releases its bus (i.e., wakes it up). The flow controller can be configured to wake up the AVP by IRQ/FIQ or other wake up events. During an AVP halt state, its clock is automatically gated by hardware.

15.1.5.2 Power-Gating

The AVP is not power-gated.

15.1.6 CPU Power Gating/Ungating Sequence

The processor’s (A15 or A9) power-gating can be triggered by either i) a direct register write to PMC or ii) interacting with the flow controller. This section describes the power-gating sequence via flow-controller interaction.

The flow controller interfaces with the Processors, PMC, and CAR to implement power-gating on/off hardware-sequence. The power-gating controller is in the PMC, while clock and reset controls are in the CAR.

15.1.6.1 CPU Power-Gating (Power OFF) Sequence

- [SW-CPU] Prepare processor (flush CPU L1$, context save, set up flow controller) for power-gating

- [SW-CPU] Execute WFI to make sure there is no outstanding processor transaction
- [HW-CPU] Assert *STANDBYWFI* to the flow controller
- [HW-Flow] Assert flow2car_*_rst
  - This is not used in power gate sequence when powergate_enable is high
  - When powergate enable is “0”, CAR uses this to assert CPU reset
- [HW-Flow] Request PMC (by asserting flow2pmc_req and associated bus) to power-off CPU
- [HW-PMC] Assert the clamp control signal. This goes to ccplex and also goes to CAR.
- [HW-CAR] Assert CPU reset and reset the CPU clock
  - [HW-CPU] Stop CPU clock
- [HW-CAR] Ack PMC to indicate CPU clock is stopped
- [HW-PMC] Power off CPU (assert CPU PG-enables)
- [HW-PMC] Clear PWRGATE_STS register bit. This is to indicate that CPU is power-gated.
- [HW-PMC] Ack to the flow controller to confirm power-off (by asserting pmc2flow_ack)
  - **Note:** After receiving ack from the PMC, the flow controller must not issue a new request to the PMC until pmc2flow_ack goes idle.
- [HW-Flow] Deassert flow2pmc_req (after seeing the Ack)
- [HW-PMC] Deassert Ack (after seeing request deassertion)
- [HW-Flow] Do not start another request until Ack goes down

15.1.6.2 CPU Power-Ungating (Power ON) Sequence

The generic CPU power-ungating (power OFF) sequence is described below. The specific details of main CPU (Cortex-A15) are described in their respective power-gating sections.

- [HW-Flow] After detecting wake up condition, request PMC to power-on CPU (by asserting flow2pmc_req and associated bus)
- [HW-PMC] Power-on the CPU (deassert CPU PG-enables)
- [HW-PMC] Set PWRGATE_STS register bit. This indicates that the CPU is power-ungated
- [HW-PMC] Ack flow controller (by asserting pmc2flow_ack) that CPU is powered on
- [HW-Flow] Deassert flow2car_*_rst to CAR
- [HW-CAR] Request CPU to ungate CPU clock (by deasserting car2ce*_cke)
- [HW-CPU] Ungate CPU clock
- [HW-CAR] Request PMC to remove CPU clamp (by asserting car2pmc_ack)
  - This is to make sure that clock gets ungated well ahead of reset deassertion
- [HW-PMC] Removes the clamping (by deasserting pmc2ce*_clamp)
  - This goes to the CPU and also to the CAR
- [HW-PMC] Ack flow controller (by deasserting pmc2flow_ack)
- [HW-CAR] Deasserts reset
- [SW-CPU] CPU fetches code from reset vector.
  - Note all of the Cortex-A15 CPUs share one reset-vector register (in EVP registers).

15.1.7 Non-CPU Power Gating/Ungating Sequence

From a software perspective, non-CPU power gating/ungating is done as part of CPU power gating/ungating request. The flow controller issues a separate request (to PMC) for power-gating CPU and non-CPU domain. The main reason is that PMC can

only sequence one power domain at a time, so the flow controller serializes requests on behalf of PMC. From the flow controller hardware perspective, the sequence for non-CPU domain power gating is similar to CPU domain power gating/ungating which is described in above sections.

There is an ARM requirement for Cortex-A15 CPUs that there must be a minimum of 16 clocks prior to clamps being deasserted on outputs from the CPU core partitions to the non-CPU partition. To address this issue, the CAR waits for CCPLEX to assert ccplex2car_pmc_cntr_ack before it deasserts car2pmc_<pgpart>_ack. CCPLEX implements the 16 CPU clock cycle wait after reset is released and clocks unstopped by the CAR, before it asserts the ack to CAR. The 16-cycle counters are implemented in the VDD_CPU power rail domain and the acks are generated from the same domain. The per-core acks are not affected by the ce0/ce1/ce2/ce3 clamps. The c0nc, c1nc, and unpower-gated VDD_CPU logic are not affected by the additional handshake logic between CCPLEX and CAR. So these partitions rely on the CAR2PMC_NONCPU_ACK_WIDTH counter for CAR to allow the PMC to release the clamp controls on these partitions.

In the CAR register space, to enable the legacy mode i.e. disable the 16-cycle counter:

- Set CLK_RST_CONTROLLER_CPU_SOFTRST_CTRL2_0_IGNORE_HW_ACK_WIDTH to 1
- Set CLK_RST_CONTROLLER_CPU_SOFTRST_CTRL2_0_IGNORE_SW_ACK_WIDTH to 0

  This will cause CPU core partition power gating FSMs in the SOC to use the CLK_RST_CONTROLLER_CPU_SOFTRST_CTRL2_0_CAR2PMC_CPU_ACK_WIDTH counter.

To enable the fix and use both the existing T3 counter and the 16-cycle counter inside CCPLEX:

- Set CLK_RST_CONTROLLER_CPU_SOFTRST_CTRL2_0_IGNORE_HW_ACK_WIDTH to 0
- Set CLK_RST_CONTROLLER_CPU_SOFTRST_CTRL2_0_IGNORE_SW_ACK_WIDTH to 0

To enable the fix and use only the 16-cycle counter inside CCPLEX but not the existing T3 counter:

- Set CLK_RST_CONTROLLER_CPU_SOFTRST_CTRL2_0_IGNORE_HW_ACK_WIDTH to 0
- Set CLK_RST_CONTROLLER_CPU_SOFTRST_CTRL2_0_IGNORE_SW_ACK_WIDTH to 1

15.1.8 CPU Rail Gating/Ungating Sequence

CPU Rail gating/ungating is only supported on Tegra K1 devices when a cluster switch is taking place or during LP0 entry when the PMIC turns the rail off.

15.1.8.1 CPU Rail Gating Constraints

The FCPU cluster cannot be rail-gated as a side-effect of the last FCPU core asserting WFI.

The ccplex2car_pmc_cntr_ack_fcpu{0-3} and DFT-related signals use tmc2clamp_pmc2c0nc_clamp as the clamp enable with the effect that the clamps are not asserted when FCCPLEX is rail gated. The tmc2clamp_pmc2c0nc_clamp signal is asserted only when the C0NC (non-CPU) partition is power gated. Since power-gating C0NC is mutually exclusive with respect to rail gating FCCPLEX, there are two constraints:

- FCCPLEX can only be rail gated by a cluster switch sequence by programming CSR_ENABLE_EXT field to 2'b10. FCCPLEX still needs to be rail gated during LP0 entry from FCPU, which results in an unavoidable window of time when the level shifters may leak when SOC and CPU rails are simultaneously being turned off by the external PMIC.
- During a cluster switch from FCCPLEX to SCCPLEX, F-C0NC should be power gated only. The FCCPLEX rail can be later turned off by software running from the SCPU:
  - Set CSR_ENABLE_EXT to 0x1
  - Trigger cluster switch
  - Toggle the CRAIL bit of APBDEV_PMC_PWRGATE_TOGGLE_0. This step will cause the correct clamps to be asserted.

When switching back to the FCPU, the CPU rail needs to be powered up using PMC controls. Flow controller RAIL_ENABLE cannot be used for this purpose.

15.1.8.2 Cluster0 RAM Re-Repair

At (cold or warm) boot power-up of the CPU rail, software powers up the CPU rail (by direct register writes to the PMC or the PMIC), and triggers RAM repair by writing to the RAM_REPAIR[REQ] register of the flow controller. After that software polls RAM_REPAIR[STS] to ensure that the repair is done.

After boot power-up of the CPU rail, whenever the CPU rail is powered up by the flow controller, the flow-controller hardware also requests cluster0 RAM re-repair, and ensures that the repair is completed.

The re-repair request and ack signals are per segment of the CPU repair chain. The request, once asserted (high), must remain high for as long as it does not receive the corresponding ack signal.

15.1.9 Cortex-A15 Debug Power Up/Down Requirements

The Cortex-A15 processor requires debug power up/down related support. The flow controller provides the following debug support.

15.1.9.1 DBGPWRUPREQ

The CoreSight can initiate power-up (for a CPU which is power-gated via the flow controller) by writing to the Cortex-A15 DBGPRCR register. As a result of this register write, A15 asserts DBGPWRUPREQ signal (a per CPU active-high level). Note, this signal can never be asserted when C0NC (or C1NC) is power-gated (since the logic generating this is inside those partitions).

The flow controller uses this signal as a wake-event (in addition to the wake events programmed in CPUx_CSR register) to wake the corresponding CPU-only. The flow controller qualifies this signal by corresponding PWRUPREQ_QUAL bit (of FLOW_DBG_QUAL register). In summary, the flow controller uses the following condition to wake up a CPU.

*CEx wake-condition = (CEx wake-event as programmed in CPUx_CSR) || (<**ccplex2flow**>**_DBGPWRUPREQx && PWRUPREQ_QUALx**)*

15.1.9.2 DBGNOPWRDWN

The CoreSight can force CPU power-down to be power-down “emulation” only by writing to the Cortex-A15 DBGPRCR register. As a result of this register write, the Cortex-A15 asserts DBGNOPPWRDWN signal (a per CPU active-high level). Note, this signal is asserted to emulate CPU-only power-gating.

The flow controller qualifies <ccplex2flow>_DBGNOPPWRDWNx signal by corresponding NOPPWRDWN_QUALx bit (of FLOW_DBG_QUAL register) and uses that as an indication of CPU power-gating without the pg_enable option.

Note that CPU power-gating is still triggered by CPUx_CSR write only (based on WFI). But regardless of CPUx_CSR register configuration, if (<>_ DBGNOPPWRDWNx && NOPPWRDWN_QUALx == TRUE), then only CPUx is power-gated without the pg_enable option.

15.1.9.3 DBGPWRDWNREQ/DBGPWRDWNACK

Cortex-A15 debug logic needs to know about power-gating requests. It also needs to make sure that power-gating starts only after it has acknowledged (so it can flush trace buffers, etc.). For this purpose, the Cortex-A15 provides per CPU DBGPWRDWNREQ/ DBGPWRDWNACK interface.

Logically, STANDBYWFIx (Cortex-A15 output) can be fed to DBGPWRDWNREQx (Cortex-A15 input). But STANDBYWFI can be asserted for many reasons, so it has to be qualified when it is asserted for CPU power-gating. The flow controller provides qualifier (PWRDWNREQ_QUALx bits of FLOW_DBQ_QUAL register) which will be set by software at the power-gating entry, and cleared by software at CPU power-ungating.

Also, if qualifier (i.e., PWRDWNREQ_QUALx) is set, then the flow controller needs to wait for DBGPWRDWNACKx (in addition to waiting for existing STANDBYWFIx condition) before starting the CPUx power-gating.

In summary, the following is needed in each ccplex cluster:

<ccplex>_ DBGPWRDWNREQx = <ccplex>_STANDBYWFIx & <flow2ccplex>_PWRDWNREQ_QUALx

The flow controller needs to wait for **<ccplex2flow>_DBGPWRDWNACKx & PWRDWNREQ_QUALx** (in addition to existing wait for STANDBYWFIx etc condition).

15.2 Flow Controller Registers

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

In the Flow Controller register descriptions, CPU refers to CPU0, and COP refers to the AVP coprocessor. CPU-LP (CPU4) has its own set of control registers. This is required because there is a case (cluster1 to cluster0 switch) where both sets of registers can be configured in parallel. Also, since CPU-LP may morph into any of the four CPUs, it is better to keep its control register separate.

The EVENTS register is replicated per core, once for COP, four times for the cores of Cluster0.

The fields and enums have the following interpretation:

- MODE defines how the flow controller logic operates; the reset value is 0x0 for CPU0 and COP (does nothing) and 0x2 for CPU1, 2, 3 (WAITEVENT). This is related to the default behavior of the reset generator.
- Enumerated values for the field MODE are defined below.

Field Mode	Enumerated Value	Description
FLOW_MODE_NONE	0	No flow control
FLOW_MODE_RUN_AND_INT	1	Keep running but generate interrupt when event conditions met (not used)
FLOW_MODE_WAITEVENT	2	Stop running until event conditions met
FLOW_MODE_WAITEVENT_AND_INT	3	Same as FLOW_MODE_WAITEVENT but generate an interrupt when resumed (not used)
FLOW_MODE_STOP_UNTIL_IRQ	4	Stop until an interrupt controller interrupt occurs
FLOW_MODE_STOP_UNTIL_IRQ_AND_INT	5	Same as FLOW_MODE_STOP_UNTIL_INT but generate another interrupt when resumed (not used)
FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ	6	Stop until event conditions met AND an interrupt controller interrupt occurs

There is no enum for 7

- Bit 31: (also known as Halt) = This bit is set to configure the wake up (of corresponding CPU) by any of the interrupts (LIC_IRQ/FIQ, or GIC_RIQ/FIQ) corresponding to that CPU. If this bit is set, then none of the other fields of this register need to be set for CPU wake. This bit is set by software and cleared by hardware when any of these interrupt is received by hardware.
- Bit 30: (also known as Wait) = This bit is set to configure the wake up (of corresponding CPU) by any of the enabled "events". The events can be enabled by setting other fields of this register (incl. *_IRQ, *_FIQ). If this bit is set, then at least one of the events needs to be enabled to generate wake event. This bit is set by software and cleared by hardware when wait event is observed by hardware.
- Bit 29: (also known as Sync) = This bit causes the NEXT Flow Controller register read to stall the processor until the wake event occurs (Used for COP. Obsolete for CPUs).

The rest of the fields define which conditions will be taken into account to restart a halted processor.

There are two types of events:

- Counted events: Decrement the field called ZERO. Flow controller resumes when this counter reaches 0
- Activity events: The flow controller resumes when the activity occurs, no counting

All conditions are disabled at reset:

- JTAG: Resume on JTAG activity
- SCLK : Resume on Nth SYSCLK cycle ticks
- X32K : Resume on Nth X32K clock input ticks
- uSEC: Resume on Nth uSEC clock ticks (uSEC = microsecond)
- mSEC: Resume on Nth mSEC clock ticks (mSEC = millisecond)
- SEC : Resume on Nth second RTC clock ticks
- X_RDY: Resume on Nth XIO.RDY Ext. IO Ready events
- SMP3[1,0]: Resume on Nth SMP.3[1,0] Semaphore set events
- XRQ_[D,C,B,A] : Resume on Nth XRQ.[D,C,B,A] External Trigger events
- [O,I]B[E,F]: Resume on Nth [O,I]B[E,F] [Outbox, Inbox] [Empty, Full] Events
- [LIC, GIC]_[IRQ,FIQ]: Resume on [LIC, GIC] [IRQ,FIQ].
- ZERO: Initialized then decremented

Note: If more than one event is enabled, the event counter will decrement based on an OR condition of enabled events

15.2.1 FLOW_CTLR_HALT_CPU_EVENTS_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:29	0x0	MODE: 0 = FLOW_MODE_NONE 1 = FLOW_MODE_RUN_AND_INT 2 = FLOW_MODE_WAITEVENT 3 = FLOW_MODE_WAITEVENT_AND_INT 4 = FLOW_MODE_STOP_UNTIL_IRQ 5 = FLOW_MODE_STOP_UNTIL_IRQ_AND_INT 6 = FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ 2 = FLOW_MODE_STOP 3 = FLOW_MODE_STOP_AND_INT 4 = FLOW_MODE_STOP_UNTIL_INT 5 = FLOW_MODE_STOP_UNTIL_INT_AND_INT 6 = FLOW_MODE_STOP_OR_INT
28	0x0	JTAG
27	0x0	SCLK
26	0x0	X32K
25	0x0	uSEC
24	0x0	MSEC
23	0x0	SEC
22	0x0	X_RDY
21	0x0	SMP31

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	LIC_IRQ
10	0x0	LIC_FIQ
9	0x0	GIC_IRQ
8	0x0	GIC_FIQ
7:0	0x0	ZERO

15.2.2 FLOW_CTLR_HALT_COP_EVENTS_0

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

Bit	Reset	Description
31:29	0x0	MODE: 0 = FLOW_MODE_NONE 1 = FLOW_MODE_RUN_AND_INT 2 = FLOW_MODE_WAITEVENT 3 = FLOW_MODE_WAITEVENT_AND_INT 4 = FLOW_MODE_STOP_UNTIL_IRQ 5 = FLOW_MODE_STOP_UNTIL_IRQ_AND_INT 6 = FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ 2 = FLOW_MODE_STOP 3 = FLOW_MODE_STOP_AND_INT 4 = FLOW_MODE_STOP_UNTIL_INT 5 = FLOW_MODE_STOP_UNTIL_INT_AND_INT 6 = FLOW_MODE_STOP_OR_INT
28	0x0	JTAG
27	0x0	SCLK
26	0x0	X32K
25	0x0	uSEC
24	0x0	MSEC
23	0x0	SEC
22	0x0	X_RDY
21	0x0	SMP31
20	0x0	SMP30

Bit	Reset	Description
19	0x0	XRQ_D
18	0x0	XRQ_C
17	0x0	XRQ_B
16	0x0	XRQ_A
15	0x0	OBE
14	0x0	OBF
13	0x0	IBE
12	0x0	IBF
11	0x0	LIC_IRQ
10	0x0	LIC_FIQ
9	0x0	GIC_IRQ
8	0x0	GIC_FIQ
7:0	0x0	ZERO

15.2.3 FLOW_CTLR_CPU_CSR_0

The CPU_CSR registers are replicated for each CPU cores. They define additional characteristics of the flow controller linked to CPU cores, especially the interaction of the flow controller with power management functions. The LP1 state is normally entered and exited via side effects of the flow controller operation.

All fields of the CPU_CSR register have an initial value of 0. The different fields are:

- PWR_STATE : Current state of the PowerGate State Machine
- WAIT_EVENT: CPU is waiting, wake up is via an event
- HALT: CPU is halted
- P2F_ACK: pmc2flow_ack signal, this is the same signal for both CPU cores
- F2P_PWRUP: flow2pmc_pwrup, this is the same signal for both CPU cores
- F2P_REQ: flow2pmc_req valid, this is the same signal for both CPU cores
- F2C_MPCORE_RST: TRUE when requesting reset of MPCore
- PWR_OFF_STS: TRUE when CPU Power-Gated is OFF by Flow Controller
- INTR_FLAG: TRUE when Interrupt is Active -- Write-1-to-Clear
- EVENT_FLAG: TRUE when Event is Active -- Write-1-to Clear
- ENABLE_EXT: If ENABLE is TRUE, then this specifies what to power off.
    - 00b: Power Gate CPU only.
    - 01b: Power Gate CPU and non-CPU (note, this option must be used only when last CPU is being power-gated).
    - 10b: Reserved.
    - 11b: PG Emulation. Note, this option is for CPU-only PG emulation except for cluster-switch case (i.e. with SWITCH_CLUSTER=1) in which emulation applies to all PG steps of source cluster. In PG emulation, PG partition is reset but not power-gated..

- IMMEDIATE_WAKE: If set, CPU is powered up immediately (without waiting for an interrupt or event). Note: If this is set then HALT_CPUx_EVENTS[MODE] should be set to 0x2 for immediate wake.
- SWITCH_CLUSTER: If set, the active cluster will be switched when all indicated CPU reach STANDBY_WFI. This bit self clears once the cluster switch sequence is completed
- WAIT_WFI_BITMAP: All cores indicated in bitmap must be in STANDBY_WFI before switching Cluster
- EVENT_ENABLE: Generates an event when the flow controller exits the halted state

- ENABLE: PowerGate Enable - Halt or Event-wait causes CPU power-gating.

Offset: 0x8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
27:24	RO	X	PWR_STATE
23	RO	X	WAIT_EVENT
22	RO	X	HALT
21	RO	X	P2F_ACK
20	RO	X	F2P_PWRUP
19	RO	X	F2P_REQ
17	RO	X	F2C_MPCORE_RST
16	RO	X	PWR_OFF_STS
15	RW	0x0	INTR_FLAG
14	RW	0x0	EVENT_FLAG
13:12	RW	0x0	ENABLE_EXT: 0 = POWERGATE_CPU_ONLY 1 = POWERGATE_BOTH_CPU_NONCPU 2 = POWERGATE_CPU_TURNOFF_CPURAIL 3 = PG_EMULATION
11:8	RW	0x0	WAIT_WFI_BITMAP
7:4	RW	0x0	WAIT_WFE_BITMAP. Not used.
3	RW	0x0	IMMEDIATE_WAKE
2	RW	0x0	SWITCH_CLUSTER
1	RW	0x0	EVENT_ENABLE
0	RW	0x0	ENABLE

15.2.4 FLOW_CTLR_COP_CSR_0

COP Control/Status for Interrupts

R/W addr=6000:700c

Offset: 0xc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxxxxxxxxxxxxxx)

Bit	Reset	Description
15	0x0	INTR_FLAG: TRUE when Interrupt is Active -- Write-1-to-Clear

15.2.5 FLOW_CTLR_XRQ_EVENTS_0

XRQ Event Detect Selector Register

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	XRQ_D7_XRQ_D0: Setting a bit to 1 enables event triggering for the corresponding bit in GPIO port D. The assertion level is determined by GPIO_INT.LVL.D. If more than one XRQ.D bit is set, the events are ORed together. The resultant event is enabled by setting the XRQ.D bit in the HALT_CPU.EVENTS or HALT_COP.EVENTS registers.
23:16	0x0	XRQ_C7_XRQ_C0: Setting a bit to 1 enables event triggering for the corresponding bit in GPIO port C. The assertion level is determined by GPIO_INT.LVL.C. If more than one XRQ.C bit is set, the events are ORed together. The resultant event is enabled by setting the XRQ.C bit in the HALT_CPU.EVENTS or HALT_COP.EVENTS registers.
15:8	0x0	XRQ_B7_XRQ_B0: Setting a bit to 1 enables event triggering for the corresponding bit in GPIO port B. The assertion level is determined by GPIO_INT.LVL.B. If more than one XRQ.B bit is set, the events are ORed together. The resultant event is enabled by setting the XRQ.B bit in the HALT_CPU.EVENTS or HALT_COP.EVENTS registers.
7:0	0x0	XRQ_A7_XRQ_A0: Setting a bit to 1 enables event triggering for the corresponding bit in GPIO port A. The assertion level is determined by GPIO_INT.LVL.A. If more than one XRQ.A bit is set, the events are ORed together. The resultant event is enabled by setting the XRQ.A bit in the HALT_CPU.EVENTS or HALT_COP.EVENTS registers.

15.2.6 FLOW_CTLR_HALT_CPU1_EVENTS_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x40000000 (0b01000000000000000000000000000000)

Bit	Reset	Description
31:29	0x2	MODE: 0 = FLOW_MODE_NONE 1 = FLOW_MODE_RUN_AND_INT 2 = FLOW_MODE_WAITEVENT 3 = FLOW_MODE_WAITEVENT_AND_INT 4 = FLOW_MODE_STOP_UNTIL_IRQ 5 = FLOW_MODE_STOP_UNTIL_IRQ_AND_INT 6 = FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ 2 = FLOW_MODE_STOP 3 = FLOW_MODE_STOP_AND_INT 4 = FLOW_MODE_STOP_UNTIL_INT 5 = FLOW_MODE_STOP_UNTIL_INT_AND_INT 6 = FLOW_MODE_STOP_OR_INT
28	0x0	JTAG
27	0x0	SCLK
26	0x0	X32K
25	0x0	uSEC
24	0x0	MSEC
23	0x0	SEC
22	0x0	X_RDY
21	0x0	SMP31
20	0x0	SMP30
19	0x0	XRQ_D

Bit	Reset	Description
18	0x0	XRQ_C
17	0x0	XRQ_B
16	0x0	XRQ_A
15	0x0	OBE
14	0x0	OBF
13	0x0	IBE
12	0x0	IBF
11	0x0	LIC_IRQ
10	0x0	LIC_FIQ
9	0x0	GIC_IRQ
8	0x0	GIC_FIQ
7:0	0x0	ZERO

15.2.7 FLOW_CTLR_CPU1_CSR_0

Offset: 0x18 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
27:24	RO	X	PWR_STATE
23	RO	X	WAIT_EVENT
22	RO	X	HALT
21	RO	X	P2F_ACK
20	RO	X	F2P_PWRUP
19	RO	X	F2P_REQ
17	RO	X	F2C_MPCORE_RST
16	RO	X	PWR_OFF_STS
15	RW	0x0	INTR_FLAG
14	RW	0x0	EVENT_FLAG
13:12	RW	0x0	ENABLE_EXT: 0 = POWERGATE_CPU_ONLY 1 = POWERGATE_BOTH_CPU_NONCPU 2 = POWERGATE_CPU_TURNOFF_CPURAIL 3 = PG_EMULATION
11:8	RW	0x0	WAIT_WFI_BITMAP
7:4	RW	0x0	WAIT_WFE_BITMAP. Not used.
3	RW	0x0	IMMEDIATE_WAKE
2	RW	0x0	SWITCH_CLUSTER
1	RW	0x0	EVENT_ENABLE

Bit	R/W	Reset	Description
0	RW	0x0	ENABLE

15.2.8 FLOW_CTLR_HALT_CPU2_EVENTS_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x40000000 (0b01000000000000000000000000000000)

Bit	Reset	Description
31:29	0x2	MODE: 0 = FLOW_MODE_NONE 1 = FLOW_MODE_RUN_AND_INT 2 = FLOW_MODE_WAITEVENT 3 = FLOW_MODE_WAITEVENT_AND_INT 4 = FLOW_MODE_STOP_UNTIL_IRQ 5 = FLOW_MODE_STOP_UNTIL_IRQ_AND_INT 6 = FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ 2 = FLOW_MODE_STOP 3 = FLOW_MODE_STOP_AND_INT 4 = FLOW_MODE_STOP_UNTIL_INT 5 = FLOW_MODE_STOP_UNTIL_INT_AND_INT 6 = FLOW_MODE_STOP_OR_INT
28	0x0	JTAG
27	0x0	SCLK
26	0x0	X32K
25	0x0	uSEC
24	0x0	MSEC
23	0x0	SEC
22	0x0	X_RDY
21	0x0	SMP31
20	0x0	SMP30
19	0x0	XRQ_D
18	0x0	XRQ_C
17	0x0	XRQ_B
16	0x0	XRQ_A
15	0x0	OBE
14	0x0	OBF
13	0x0	IBE
12	0x0	IBF
11	0X0	LIC_IRQ
10	0X0	LIC_FIQ
9	0X0	GIC_IRQ
8	0X0	GIC_FIQ
7:0	0X0	ZERO

15.2.9 FLOW_CTLR_CPU2_CSR_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
27:24	RO	X	PWR_STATE
23	RO	X	WAIT_EVENT
22	RO	X	HALT
21	RO	X	P2F_ACK
20	RO	X	F2P_PWRUP
19	RO	X	F2P_REQ
17	RO	X	F2C_MPCORE_RST
16	RO	X	PWR_OFF_STS
15	RW	0x0	INTR_FLAG
14	RW	0x0	EVENT_FLAG
13:12	RW	0x0	ENABLE_EXT: 0 = POWERGATE_CPU_ONLY 1 = POWERGATE_BOTH_CPU_NONCPU 2 = POWERGATE_CPU_TURNOFF_CPURAIL 3 = PG_EMULATION
11:8	RW	0x0	WAIT_WFI_BITMAP
7:4	RW	0x0	WAIT_WFE_BITMAP. Not used.
3	RW	0x0	IMMEDIATE_WAKE
2	RW	0x0	SWITCH_CLUSTER
1	RW	0x0	EVENT_ENABLE
0	RW	0x0	ENABLE

15.2.10 FLOW_CTLR_HALT_CPU3_EVENTS_0

Offset: 0x24 | Read/Write: R/W | Reset: 0x40000000 (0b01000000000000000000000000000000)

Bit	Reset	Description
31:29	0x2	MODE: 0 = FLOW_MODE_NONE 1 = FLOW_MODE_RUN_AND_INT 2 = FLOW_MODE_WAITEVENT 3 = FLOW_MODE_WAITEVENT_AND_INT 4 = FLOW_MODE_STOP_UNTIL_IRQ 5 = FLOW_MODE_STOP_UNTIL_IRQ_AND_INT 6 = FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ 2 = FLOW_MODE_STOP 3 = FLOW_MODE_STOP_AND_INT 4 = FLOW_MODE_STOP_UNTIL_INT 5 = FLOW_MODE_STOP_UNTIL_INT_AND_INT 6 = FLOW_MODE_STOP_OR_INT
28	0x0	JTAG
27	0x0	SCLK

Bit	Reset	Description
26	0x0	X32K
25	0x0	uSEC
24	0x0	MSEC
23	0x0	SEC
22	0x0	X_RDY
21	0x0	SMP31
20	0x0	SMP30
19	0x0	XRQ_D
18	0x0	XRQ_C
17	0x0	XRQ_B
16	0x0	XRQ_A
15	0x0	OBE
14	0x0	OBF
13	0x0	IBE
12	0x0	IBF
11	0X0	LIC_IRQ
10	0X0	LIC_FIQ
9	0X0	GIC_IRQ
8	0X0	GIC_FIQ
7:0	0X0	ZERO

15.2.11 FLOW_CTLR_CPU3_CSR_0

Offset: 0x28 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
27:24	RO	X	PWR_STATE
23	RO	X	WAIT_EVENT
22	RO	X	HALT
21	RO	X	P2F_ACK
20	RO	X	F2P_PWRUP
19	RO	X	F2P_REQ
17	RO	X	F2C_MPCORE_RST
16	RO	X	PWR_OFF_STS
15	RW	0x0	INTR_FLAG
14	RW	0x0	EVENT_FLAG

Bit	R/W	Reset	Description
13:12	RW	0x0	ENABLE_EXT: 0 = POWERGATE_CPU_ONLY 1 = POWERGATE_BOTH_CPU_NONCPU 2 = POWERGATE_CPU_TURNOFF_CPURAIL 3 = PG_EMULATION
11:8	RW	0x0	WAIT_WFI_BITMAP
7:4	RW	0x0	WAIT_WFE_BITMAP. Not used.
3	RW	0x0	IMMEDIATE_WAKE
2	RW	0x0	SWITCH_CLUSTER
1	RW	0x0	EVENT_ENABLE
0	RW	0x0	ENABLE

15.2.12 FLOW_CTLR_CLUSTER_CONTROL_0

Offset: 0x2c │ Read/Write: R/W │ Reset: 0x04004000 (0b00000100000000000001000000xxxxxxx0)

Bit	Reset	Description
31:20	0x40	POST_SWITCH_DELAY: Two delays on top of the built in pipeline delay when switching cluster, in clock cycles
19:8	0x40	PRE_SWITCH_DELAY
0	0x0	ACTIVE: The Active field can be written, this should only be done by the COP after making sure all clusters are inactive. Directly writing the Active field will otherwise most probably result in serious errors as there are no interlock in hardware as when the flow controller is used to control the switch together with the use of WFI on the currently inactive cluster 0 = G 1 = LP

15.2.13 FLOW_CTLR_HALT_COP1_EVENTS_0

Offset: 0x30 │ Read/Write: R/W │ Reset: 0x40000000 (0b01000000000000000000000000000000)

Bit	Reset	Description
31:29	0x2	MODE: 0 = FLOW_MODE_NONE 1 = FLOW_MODE_RUN_AND_INT 2 = FLOW_MODE_WAITEVENT 3 = FLOW_MODE_WAITEVENT_AND_INT 4 = FLOW_MODE_STOP_UNTIL_IRQ 5 = FLOW_MODE_STOP_UNTIL_IRQ_AND_INT 6 = FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ 2 = FLOW_MODE_STOP 3 = FLOW_MODE_STOP_AND_INT 4 = FLOW_MODE_STOP_UNTIL_INT 5 = FLOW_MODE_STOP_UNTIL_INT_AND_INT 6 = FLOW_MODE_STOP_OR_INT
28	0x0	JTAG
27	0x0	SCLK
26	0x0	X32K
25	0x0	uSEC

Bit	Reset	Description
24	0x0	MSEC
23	0x0	SEC
22	0x0	X_RDY
21	0x0	SMP31
20	0x0	SMP30
19	0x0	XRQ_D
18	0x0	XRQ_C
17	0x0	XRQ_B
16	0x0	XRQ_A
15	0x0	OBE
14	0x0	OBF
13	0x0	IBE
12	0x0	IBF
11	0x0	LIC_IRQ
10	0x0	LIC_FIQ
9	0x0	GIC_IRQ
8	0x0	GIC_FIQ
7:0	0x0	ZERO

15.2.14 FLOW_CTLR_COP1_CSR_0

Offset: 0x34 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00xxxxx0xxx00x00)

Bit	R/W	Reset	Description
27:24	RO	X	PWR_STATE
23	RO	X	WAIT_EVENT
22	RO	X	HALT
21	RO	X	P2F_ACK
20	RO	X	F2P_PWRUP
19	RO	X	F2P_REQ
17	RO	X	F2C_MPCORE_RST
16	RO	X	PWR_OFF_STS
15	RW	0x0	INTR_FLAG
14	RW	0x0	EVENT_FLAG
8	RW	0x0	WAIT_WFI_BITMAP
4	RW	0x0	WAIT_WFE_BITMAP. Not used.

Bit	R/W	Reset	Description
3	RW	0x0	IMMEDIATE_WAKE
1	RW	0x0	EVENT_ENABLE
0	RW	0x0	ENABLE

15.2.15 FLOW_CTLR_CPU_PWR_CSR_0

Offset: 0x38 │ Read/Write: R/W │ Reset: 0x00000100 (0bxxxxxxxxxxxxxxxxxxxxxxxx100000000)

Bit	Reset	Description
8	0x1	CPU_RG_CFG: If this bit is set, then the flow controller would initiate last-CPU PG (along with CPU RG (rail-gating) or non-CPU PG) only when other 3 CPUs are already power-gated. This is primarily for debug purposes. 0 = DISABLE 1 = ENABLE
7:6	0x0	C1NC_STS: Hardware updates this register based on the current status of the C1NC partition. Before requesting a power-off (or on) request, flow controller updates this status to "in the process" status. Note that the flow controller requests C1NC power on/off based on this register value or PMC pwrgate-status of C1NC depending upon the USE_FLOW_STS register. For debug purposes, this can be written by software. 0 = PARTITION_OFF 1 = PG_IN_PROGRESS 2 = PU_IN_PROGRESS 3 = PARTITION_ON
5:4	0x0	C0NC_STS: hardware updates this register based on the current status of the C0NC partition. Before requesting a power-off (or on) request, flow controller updates this status to "in the process" status. Note, flow controller requests C0NC power on/off based on this register value or PMC pwrgate-status of C0NC depending upon USE_FLOW_STS register. For debug purposes, this can be written by software. 0 = PARTITION_OFF 1 = PG_IN_PROGRESS 2 = PU_IN_PROGRESS 3 = PARTITION_ON
3	0x0	USE_FLOW_STS:If this is set (=1), then the flow controller will use C0NC_STS/ C1NC_STS/RAIL_STS bits to determine whether to power-gate C0NC/C1NC/CRAIL domains. If this bit is clear (=0), then the flow controller will use the PMC pwrgate-status of C0NC/C1NC/CRAIL to determine whether to power-gate the C0NC (and C1NC) domains. In either case, the C0NC_STS/C1NC_STS/RAIL_STS fields are updated in the same way.
2:1	0x0	RAIL_STS: Hardware updates this register based on the current status of the CPU-rail. Before requesting a power-off (or on) request, flow controller updates this status to "in the process" status. Note, flow controller requests CPU-RAIL power on/off based on this register value or PMC pwrgate-status of CRAIL depending upon USE_FLOW_STS register. For debug purposes, this can be written by software. 0 = RAIL_OFF 1 = RG_IN_PROGRESS 2 = RU_IN_PROGRESS 3 = RAIL_ON
0	0x0	RAIL_ENABLE: CPU rail power-on request. Software sets this to request CPU rail pwr-on by flow-controller interaction. If rail is already on, writing to this is ignored. Hardware clears this bit when CPU power rail in turned on.

15.2.16 FLOW_CTLR_MPID_0

Offset: 0x3c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	CPU_ID: CPU-ID of C1 CPU. Software programs this field before cluster0 -> cluster1 switch. This ID is provided to MPCore_LP which is what is read when the OS reads MPIDR.

15.2.17 FLOW_CTLR_RAM_REPAIR_0

Offset: 0x40 │ Read/Write: R/W │ Reset: 0x00000004 (0bxxxxxxxxxxxxxxxx00000000xxxx01x0)

Bit	R/W	Reset	Description
23:16	RO	X	DBG_STS: Repair done (acknowledge) from repair logic.
15:8	RW	0x0	DBG_REQ: Repair request for individual segments.
3	RW	0x0	DBG_EN: Debug enable to be able to repair of individual segments of the cluster0 repair chain 0 = DISABLE 1 = ENABLE
2	RW	0x1	BYPASS_EN: RAM repair bypass enable. 0: Flow-controller requests RAM-repair at CPU RAIL power-up. 1: Flow-controller does not request RAM-repair at CPU RAIL power-up 0 = DISABLE 1 = ENABLE
1	RO	X	STS: Indicates Cluster repair chain status. hardware sets this to '1' when repair of all segments is done. hardware clears this to '0' when RAM repair is requested.
0	RW	0x0	REQ: Cluster0 RAM repair request. Software sets this bit to '1' to initiate RAM repair request to all segments in parallel. Hardware clears this bit when software triggered repair is done (i.e., STS=1) 0 = DISABLE 1 = ENABLE

15.2.18 FLOW_CTLR_FLOW_DBG_SEL_0

Offset: 0x44 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxx0000)

Bit	R/W	Reset	Description
19:18	RO	X	RG_PWR_STATE: Current state of Rail Gate state machine
17:16	RO	X	RU_PWR_STATE: Current state of Rail Ungate state machine
15:14	RO	X	NC_PG_PWR_STATE: Current state of Non CPU Power Gate state machine
13:12	RO	X	NC_PU_PWR_STATE: Current state of Non CPU Power Ungate state machine

Bit	R/W	Reset	Description
11:8	RW	0x0	CNT1_SEL: Activity selection which would be counted by CNT1. **Note** power/rail activities are counted only when power/rail gating is via flow controller (and not direct PMC register write). 0: Idle (counter holds its current value) 1: Increment by 1 every time CPU0 is power gated 2: Increment by 1 every time CPU1 is power gated 3: Increment by 1 every time CPU2 is power gated 4: Increment by 1 every time CPU3 is power gated 5: Increment by 1 every time and of CPU0/1/2 or 3 is power-gated 6: Increment by 1 every time CPU-LP is power gated 7: Increment by 1 every time Ccluster0 non-CPU is power gated 8: Increment by 1 every time Cluster1 non-CPU is power gated 9: Increment by 1 every time CPU RAIL is powered off 10: Increment by 1 every time cluser0 is switched to cluster1 11: Increment by 1 every time cluser1 is switched to cluster0 0 = IDLE 1 = CPU0_PG 2 = CPU1_PG 3 = CPU2_PG 4 = CPU3_PG 5 = CPU0123_PG 6 = CPULP_PG 7 = C0NC_PG 8 = C1NC_PG 9 = CRAIL_OFF 10 = C0_TO_C1_SWITCH 11 = C1_TO_C0_SWITCH
3:0	RW	0x0	CNT0_SEL: Activity selection which would be counted by CNT0. **Note:** Power/rail activities are counted only when power/rail gating is via the flow controller (and not direct PMC register write). 0: Idle (counter holds its current value) 1: Increment by 1 every time CPU0 is power gated 2: Increment by 1 every time CPU1 is power gated 3: Increment by 1 every time CPU2 is power gated 4: Increment by 1 every time CPU3 is power gated 5: Increment by 1 every time and of CPU0/1/2 or 3 is power-gated 6: Increment by 1 every time CPU-LP is power gated 7: Increment by 1 every time Ccluster0 non-CPU is power gated 8: Increment by 1 every time Cluster1 non-CPU is power gated 9: Increment by 1 every time CPU RAIL is powered off 10: Increment by 1 every time cluser0 is switched to cluster1 11: Increment by 1 every time cluser1 is switched to cluster0 0 = IDLE 1 = CPU0_PG 2 = CPU1_PG 3 = CPU2_PG 4 = CPU3_PG 5 = CPU0123_PG 6 = CPULP_PG 7 = C0NC_PG 8 = C1NC_PG 9 = CRAIL_OFF 10 = C0_TO_C1_SWITCH 11 = C1_TO_C0_SWITCH

15.2.19 FLOW_CTLR_FLOW_DBG_CNT0_0

Offset: 0x48 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE: Activity count value (updated by hardware based on CNT0_SEL). This register can be written by software to be able to clear it.

15.2.20 FLOW_CTLR_FLOW_DBG_CNT1_0

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

Bit	Reset	Description
31:0	0x0	VALUE: Activity count value (updated by hardware based on CNT1_SEL). This register can be written by software to be able to clear it.

15.2.21 FLOW_CTLR_FLOW_DBG_QUAL_0

Offset: 0x50 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx0000xxx00000xxx00000xxx00000)

Bit	Reset	Description
27:24	0x0	AXICIF_CG_DIS: CCPLEX AXICIF/MCCIF clock gating disable.
20:16	0x0	PWRUPREQ_QUAL: CPU DBGPWRUPREQ qualifier.
12:8	0x0	NOPWRDWN_QUAL: CPU DBGNOPPWRDN qualifier.
4:0	0x0	PWRDNREQ_QUAL: CPU DBGPWRDNREQ qualifier.

15.2.22 FLOW_CTLR_FLOW_CTLR_SPARE_0

Offset: 0x54 | Read/Write: R/W | Reset: 0xffff0000 (0b11111111111111110000000000000000)

Bit	Reset	Description
31:16	0xffff	SPARE_HI
15:0	0x0	SPARE_LO

15.2.23 FLOW_CTLR_RAM_REPAIR_CLUSTER1_0

Offset: 0x58 | Read/Write: R/W | Reset: 0x00XX000X (0bxxxxxxxxxxxxxxxxx00000000xxxx0xx0)

Bit	R/W	Reset	Description
23:16	RO	X	DBG_STS: Repair done (acknowledge) from repair logic. There is one segment. Bits 23:17 are unused. Reads to these bits will always return 0s. Bits 23:18: unused Bit 17 - sl2 Bit 16 - scpu
15:8	RW	0x0	DBG_REQ: Repair request for individual segments. There is one segment. Bits 15:9 are unused. Reads to these bits will always return 0s Bits 15:10: unused Bit 9 - sl2 Bit 8 - scpu

Bit	R/W	Reset	Description
3	RW	0x0	DBG_EN: Debug enable to be able to repair of individual segments of the cluster0 repair chain 0 = DISABLE 1 = ENABLE
1	RO	X	STS: Indicates Cluster repair chain status. Hardware sets this to '1' when repair of all segments is done. Hardware clears this to '0' when RAM repair is requested
0	RW	0x0	REQ: Cluster1 RAM repair request. Software sets this bit to '1' to initiate RAM repair request to all segments in parallel. Hardware clears this bit when software-triggered repair is done (i.e., STS=1) 0 = DISABLE 1 = ENABLE

16.0 MEMORY CONTROLLER

Tegra® K1 devices feature a hybrid 2x32-bit / 1x64-bit memory controller. These are interleaved to provide high performance with the load shared across both channels.

The Tegra K1 memory controller (MC) handles memory requests from internal clients and arbitrates among them to allocate memory bandwidth for DDR3L and LPDDR3 SDRAMs.

The Tegra K1 memory controller (MC) handles memory requests from internal clients and arbitrates among them to allocate memory bandwidth for DDR3L and LPDDR3 SDRAMs. The external memory controller (EMC) communicates with external DDR3L and LPDDR3 devices.

Key features in the Tegra K1 memory controller include:

- Enhanced arbiter design for higher memory efficiency
- System Memory Management Unit (SMMU)/Translation Unit (TU) for virtual to physical address mapping for any device
- Support for low-voltage DDR3 and LPDDR3 SDRAMs
- 8 burst transfers per transaction (BL8)
- Support for two DRAM ranks of unequal device densities
- Operates in either single x32 or single x64 configuration
- Variable transaction sizes based on the requests from the clients (one 64-byte transaction with variable dimensions, two 32-byte transactions with variable dimensions, etc.)
- x32 sub-partitions support for x64 configuration: additional address bits allow targeting different columns of each sub-partition
- QUSE functionality for qualification of the tristatable DQS clock in SDRAMs

The memory interface speed varies with memory type and the specific Tegra K1 SKU, so is not stated in this document.

16.1 Memory Controller Architecture

The memory controller architecturally consists of the following parts:

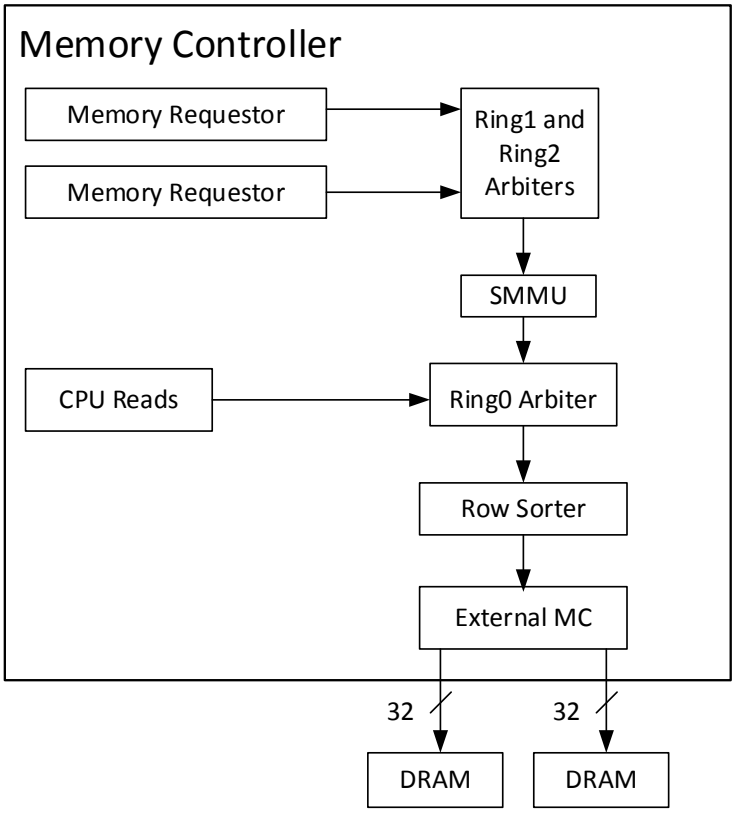
- Arbitration Domains (ADs), which can handle a single request or response per clock from a group of clients. Typically, a system has a single Arbitration Domain, but an implementation may divide the client space into multiple Arbitration Domains to increase the effective system bandwidth. Multiple Traffic Classes within a single Arbitration Domain and Protocol Arbiter are allowed.
- Protocol Arbiters (PAs), 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 entities such as configuration registers which are shared across the Memory Subsystem.
- Write CAMs (WCAMs), which improves performance and throughput for PCIe ordered clients (PCIe, SATA, HDA, and USB3), CPU writes, and CPU copies.
- Memory Controller Client Interface (MCCIF), which provides a standardized interface for access to the Memory Controller.
- Translation Unit, which handles virtual-to-physical address translation, aperture decode, physical address security checks, and protocol arbiter-specific decodes (such as external DRAM address decodes).

The Tegra K1 device supports a single physical channel of memory interface, which operates in either single 1x32 or hybrid 2x32 / 1x64 configurations. There is an alternate path through the memory controller to access IRAM via AHB re-direction.

Figure 29 below is a simplified view of how memory requests are arbitrated in the Tegra K1 MSS. Memory client requests are first arbitrated through a sequence of ring arbiters which perform a type of round-robin arbitration. There are three ring arbiters referred to as ring0, ring1, and ring2. Ring1 arbiter clients are the ISO clients (display and camera) and the winner of the ring2 arbiter. Each ring has a rate control mechanism referred to as Priority Tier Snap Arbiter (PTSA). The client's bandwidth guarantee is specified by the PTSA rate (also referred to as "DDA").

The block labelled "Row Sorter" is a pending request buffer (it sorts requests by the DRAM row that it refers to). This row sorter is made up of many "bank queues" which hold the requests made to the same DRAM bank/row. The number of requests pending in the row sorter can affect whether ring1 or ring2 arbiters are throttled (slowed down) based on thresholds.

Figure 29: Memory Request Arbitration



16.2 Hardware Features

16.2.1 DRAM Protocol Arbiter Features

- Hybrid 2x32-bit / 1x64-bit data bus
  - 4 chip selects
  - 4 individually controllable clock-enables
  - 4 individually controllable ODTs (DDR3L)
  - Operates in either single x32 or hybrid 2x32 / 1x64 configuration
  - Supports per-byte data masks
- Each chip select may support different sizes and geometries

  - Size of Rank 0 must be larger than or equal to Rank 1
  - 2 or 3 bank bits
  - Column width: 9 to 12 bits
  - Row width: 12 to 16 bits
- Per-byte data masks
- Support for BL8
- Data transfer minimization:
  - DDR3 burst chop support
- Discrete LPDDR3 and DDR3 devices up to 933 MHz
- Support for 1T and 2T
- Support for low-power modes:
  - Software controllable entry/exit from: self-refresh, power down, deep power down
  - Hardware dynamic entry/exit from: power-down, self-refresh
  - Support for intermittent or disabled DLL
  - Disable unused address/command taps based on whether in POP or Discrete mode.
  - Pads use DPD mode during idle periods
- For Tegra K1 devices, trims can be set/adjusted per-byte AND per-chip-select. The PVT-compensated value can still be used and a PVT/frequency-compensated adjustment can be added to it (1/16, 1/8, 3/16 of a cycle) or a non-PVT-offset can be added.
- Support for x32, x16, or x8 chips attached to the channel
  - DQ/DQS swizzling: MRR_BYTESEL need to match byte swizzling.
- Deadline-based arbitration with a latency allowance that can be specified per-client, and under some circumstances, dynamically adjusted for a given client.

16.2.2 Memory Crossbar

The Memory Crossbar (MXB) is in charge of adapting Arbitration Domains (ADs) to Protocol Arbiters (PAs). In the simplest system with a single AD and PA running off of a single clock, the MXB is a simple pass-through or does not exist. The Tegra K1 MXB has one AD and two PAs, one for DRAM and the other for requests redirected to the AHB. The MXB handles conflict arbitration between responses from multiple PAs to the same AD.

16.2.3 Memory Sub-Partitions

The Tegra K1 Memory Controller implements sub-partitions. The 64B memory atom is divided into two sectors: one sector containing bytes 0 to 31 and the other containing bytes 32 to 63. Each sector is stored in one of the two DRAM channels. With three additional column address bits for the second sector, clients can make 64B requests of different shapes while retaining the bandwidth and power efficiency of a 64B memory atom. This feature also allows the CPU to request that one of the two 32B sectors in a 64B atom be returned first.

The sub-partition mechanism is used by the GPU, display clients (for rotation), VIC (for rotation), MSENC (in x,y flip mode), and the CPU (for critical-word-first). Clients that choose not to use the sub-partition feature set the extra column address bits to the same values as the other channel. They see the DRAM as a simple 64B memory system. The sub-partition feature is only beneficial in systems with a 64b memory. In systems with 32b memory, client addressing and request granularity are unchanged, but the MC treats each 32B sector as an independent 32B atom.

See the “Sub-Partitions” subsection for more information.

16.2.4 Global Resources

The Memory Subsystem also contains some global resources not owned by a particular AD or PA:

- A combined SMMU/TU
- Configuration registers and statistics counters

16.2.5 Supported Page Sizes

The MSS considers the following uses of pages and page size:

- DRAM page sizes from 1KB to 4KB.
- OS page: 4KB for all operating systems
- Row Sorter page: 1KB

16.3 Software Features

The majority of software interaction with the memory subsystem involves:

- Initial configuration
- Power management
- Device management
- Translation management
- ISO client bandwidth allocation
- Statistics and debugging

16.3.1 Initial Configuration

The details for configuring the Memory Subsystem (MSS) can be broken down into the following:

- Global Memory Subsystem configuration
- Arbitration Domain configuration
- Protocol Arbiter configuration
- Client configuration

> **Note:** This configuration must be performed as part of the chip-wide boot sequence

16.3.1.1 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.

The Arbitration Domains operate off of a unified arbitration clock. This clock is further divided-down to produce a clock-rate independent clock for counting latency intervals across the Memory System (also known as “ticks”). This divide-down should be programmed to a constant interval at initialization (and any other clock-rate change). A 30 ns interval is suggested since it is an even divide-down of many common DRAM clocks, but any convenient granularity may be chosen.

There are several global memory system tuning options that tend to shape the memory performance. Some of these can be set statically, at initialization time, others depend on clock frequencies. For example, with SMMU configuration, the SMMU requires software to set up and maintain page tables in memory, enable translation for clients, and assign clients to address space identifiers.

16.3.1.2 Arbitration Domain Configuration

Physical implementation decisions (such as the mapping of clients to partition clients) and default client bandwidth allocations require configuration options in the Arbitration Domain.

16.3.1.3 Protocol Arbiter Configuration

Each Protocol Arbiter has its own requirements for initialization. Timing parameters specific to the Arbiter, the DRAM and the current operating clock speed have to be written into Protocol Arbiter configuration registers. The type and geometry of the attached DRAM also have to be programmed. Afterwards, the DRAM will require a set of initialization cycles to be issued.

Device/Rank Geometry

DRAM devices come in many sizes, widths, etc. The arbiter must be programmed to drive the correct combination of address bits, data bits, and protocol.

Device Timing and Arbiter Timing

A DRAM arbiter has configuration related to timing parameters specific to the attached device. The JEDEC timing specifications for the device have to be converted from nanoseconds to cycle counts. The controller must also be programmed with the cycles per tick number for their particular clock domain (see also the subsection on “Global Memory Subsystem Configuration” below).

Other Arbiter Parameters

Other parameters related to arbitration that need to be configured include:

- overall number of requests outstanding
- which clients are considered isochronous
- which clients participate in power-saving hysteresis operations

Device/Rank Initialization and Calibration Cycles

In particular, PAs 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 the 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.

16.3.1.4 Client Configuration

The Priority Tier Snap Arbiters normally are configured dynamically to provide bandwidth guarantees to ISO clients. However, for certain clients the PTSA settings are programmed statically (for example, when the PTSA is used as a timeout). For these clients, the PTSA settings are part of the initial configuration.

Latency allowance may also be configured dynamically for certain clients. Latency allowance should be programmed to default values during initial configuration.

16.3.2 Power Management

The primary forms of system power management involve transition to and from low-power DRAM states and management of clock frequency.

16.3.2.1 Low-Power States

The DRAM controller and DRAMs can operate in a number of low-power states. The 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/rank – it is possible for one device/rank 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.

16.3.2.2 Request Dribble

The “OSIdle” use cases (with or without display active) are important cases of active power management. Display and audio clients can be prone to “dribbling”, that is, sending isolated requests at infrequent intervals.

There are two main causes of dribbling:

- Isochronous dribble, caused by a low periodic request rate. This can be (and should be) managed in the Display client by using hysteresis (high and low watermarks) on the request buffer to group requests into bursts.
- Clock crossing dribble, which is caused by a large disparity between the client clock and the Arbitration Domain clock. If the client’s clock is slow (which it often is for Display), individual requests will have several idle cycles between each request. In general, this is unavoidable by the client.

It is difficult for the Protocol Arbiter to handle the second type of dribble when it is under load, because it cannot “see” a large enough window of requests to group the traffic efficiently. Low-data-rate isochronous clients should have client-side request hysteresis to handle isochronous dribble. The Protocol Arbiter can handle clock-crossing dribble when it is under load, because the delays are smaller and it will schedule other traffic preferentially while waiting for the dribbling traffic to accumulate. When in the OSIdle use case, the Protocol Arbiter will not have enough traffic to keep the client dribble from becoming DRAM dribble (isolated requests that open and close the DRAM pages more often than necessary).

To manage DRAM dribble, the controller adds the concept of the “*hysteresis hold-off*” (also known as iso holdoff), which is a per-controller state bit. When the hold-off is active, the controller is held off from issuing activate commands to open new DRAM pages. Incoming requests will back up in the Protocol Arbiter’s row sorter and store buffer until the hold-off is released.

The hold-off is enabled whenever an individual device goes idle due to lack of work. It is released automatically whenever a latency timer expires, or the Protocol Arbiter stalls due to fullness. The net effect of the hold-off is to group *all* requests for the DRAM into activity bursts.

The hold-off causes a delay in requests transitioning from a memory-idle state. This can affect some traffic in a negative manner – principally requests that are low-latency like the CPU. To keep the hold-off from impacting these requests, software has a per-module configuration bit that disables the hold-off for certain known-latency-sensitive clients. Whenever a request with the holdoff-disable bit set enters the Protocol Arbiter, the hold-off is released. The holdoff is enabled whenever an individual device/rank goes idle due to lack of pending requests. The holdoff is released automatically whenever a request expires, the Protocol Arbiter stalls due to fullness, a request from a non-hysteresis client is received, or the bank-queue length reaches a programmable threshold. The net effect of the holdoff is to group requests for the DRAM into activity bursts.

16.3.2.3 Clock Frequency Scaling

A final aspect of power management is scaling the Memory System clocks. In principle, there are two sets of clocks that can be controlled:

- The 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.

Tegra K1 devices have a single clock domain for the Arbitration Domain and the Protocol Arbiter: mcclk. The command controller runs on a second clock called emcclk, which matches the frequency of the command bus on the DRAMs. Mcclk can be configured to run at 1x, 1/2x, or 1/4x of emcclk. Mcclk and emcclk are phase-aligned.

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. 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.

16.3.3 Device Management

The EMC is primarily responsible for DRAM device management (ZQCAL, thermal management, etc.).

The exception to this is that the MC schedules refresh commands to the DRAM.

16.3.4 Translation Management

Surface allocation for most Tegra K1 clients is straightforward: software allocates the surface out of the device's virtual space, and then backs the mappings with physical pages. The mapping between virtual and physical addresses is described using PTE and PDEs in the SMMU's page table.

GPU surfaces have their own virtual address space that is translated by the GMMU within the GPU. The GMMU either translates requests into physical addresses that bypass the SMMU or into a second virtual address space that is then translated into physical addresses by the SMMU.

16.3.5 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.

16.3.6 Coherency and Ordering

For coherence and ordering:

1. The MSS does not guarantee the coherency between different clients if the write-response tag has not been provided back to the clients.
2. The MSS guarantees that if a write is sent with a write-response tag and that tag is returned to the client, no other read or write issued after the response is received from anywhere in the system can pass that write.
3. The MSS guarantees that within a single client, write responses will complete in-order.
4. For PCIe ordered clients, the MSS guarantees that all writes will complete in-order.

For the most part, the memory system is fire-and-forget. Software however needs to be involved in the configuration, translation, and power management services as described above.

16.4 Software Interfaces

The primary software interface for configuration is via the APB bus registers.

- Initial configuration
- Power management
- Device/rank management
- Translation management
- ISO client bandwidth allocation (per use case)
- Statistics and debugging

16.4.1 Boot

Tegra K1 devices have a two-stage boot:

- The Boot ROM is responsible for getting the chip out of reset to where driver-level software can take over.
- The Boot Loader or the OS-recovery code is the portion of the driver responsible for completing the boot process to the general-purpose OS.

The Boot ROM is an actual ROM built into the chip, and the Boot Loader and OS recovery section are externally loaded code. The Boot Loader is only used for Cold boot; Warm boot utilizes operating-system recovery code.

Tegra K1 devices can “boot” in two ways: “Warm boot” or “Cold boot”. Cold boot is defined as booting from a fully powered-off or fully reset state. Warm boot is defined as recovery from the lowest-power state (LP0), where the external DRAM has been held in self-refresh mode while the Tegra K1 core was powered off. Warm boot use case occurs much more often; for example, it is the use case hit whenever a SmartPhone wakes from Standby to answer a call.

16.4.1.1 IRAM Use

Tegra K1 device contain a protocol arbiter for accessing IRAM (ARC). This path is used only during boot.

The use of this path does not require the SMMU to be set as the IRAM addresses are expected to be physical addresses. The MC redirects requests to IRAM before the SMMU translation.

This is an additional step in the boot process for these scenarios. ARC_CLK_OVR_ON must be enabled to use the ARC path. Once all IRAM accesses are done for boot, ARC_CLK_OVR_ON should be disabled.

16.4.1.2 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. Fetch the BCT values from the storage device and move them to IRAM using AHB redirection.
2. Configure the PLL used by the MC/EMC based on BCT values
3. Wait for PLLs to lock
4. Program necessary always-on static pad configurations in the PMC
5. Configure memory clocks based on BCT values
6. Enable the MC/EMC clocks
7. Release memory subsystem resets
8. Program other necessary pad/pad-macro configuration
9. Program initial configuration for the MC/EMC based on BCT values
10. Perform the DRAM initialization sequence. This sequence is specific to the DRAM protocol used and varies depending on the requirements of a particular device. For example, a DDR3 sequence goes as follows:
    a. Apply power to the device
    b. Wait until voltage is stable
    c. Assert DRAM reset
    d. Wait 200 µs
    e. Deassert DRAM reset
    f. Apply stable clocks

g. Wait 500 µs
h. Assert and hold CKE high
i. Precharge all banks
j. Send 2 Auto-refresh commands
k. Write to Mode register 0, 1, 2, 3
l. Issue ZQ Calibration command
m. Wait for ZQ calibration and DRAM DLL lock time to complete

11. Enable power-saving features such as clock stop, active power down, dynamic-self-refresh
12. Start periodic sequences such as auto refresh, ZQ calibration, auto-calibration
13. Release the memory-initialization hardware lock[1]

After this, the external DRAM and Memory Subsystem are ready for use.

16.4.1.3 Warm Boot

The Warm Boot sequence is very similar to the Cold Boot sequence; many of the same steps are performed. One difference is that instead of powering on the DRAM, it is awakened out of SelfRefresh. Another difference is the configuration data for Boot ROM portion of MC/EMC initialization is stored in the Power Management Controller (PMC) Scratch registers.

The PMC Scratch registers are in the Always-On partition, and are powered during LP0. The power and area limitations of this implementation restricts the number of PMC Scratch registers available; there are a smaller number of PMC Scratch Registers than the number of bits needed to fully configure the MC and EMC. The Scratch Registers are only used to store the minimum required configuration to get to working DRAM. The Boot ROM may not be able to Warm boot to full POR speed using only the Scratch Register data; if this limitation is confirmed, the 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.

The Warm boot code provided by the MSS includes algorithms for deriving the approximate MC arbiter performance parameters from the EMC timing parameters. The EMC has provided generation code for the packing/unpacking calls for the scratch registers to ease the software maintenance burden.

The Warm boot Boot ROM sequence:

1. Configure memory PLLs and memory clocks based on Scratch values
2. Wait for PLLs to lock
3. Configure memory clocks based on scratch values
4. Enable the MC/EMC clocks
5. Release memory subsystem resets. Program any necessary static pad configuration that was powered-off in LP0
6. Program any necessary static pad configuration that was powered-off in LP0.
7. Program initial configuration for the MC/EMC based on Scratch values
8. 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

[1] AHB_ARBITRATION_XBAR_CTRL.MEM_INIT_DONE

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

9. Enable power saving features such as clock stop, active power down, dynamic-self-refresh
10. Start periodic sequences such as auto refresh, ZQ calibration, auto-calibration
11. 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.

16.4.2 Security Apertures

To secure the MC registers, the CPU complex's page table entries for the MC register space must be marked as secured. Then to access the registers securely, the CPU must be switched into secure mode. For more details about these CPU operations, please read the appropriate ARM Architecture Reference Manual.

To secure a region of EMEM, the page table entries for that region must be marked as secured. Then the MC registers must be configured for the secured region, using the CPU 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 CPU must be in secure mode to access the described region of memory:

- Video Protection region
- Security Coprocessor Secure region

For more information on handling security violations, see the subsection "MC Interrupts" below.

16.4.3 Software Client-Groupings

The Memory Subsystem inherently implements a hardware mapping of memory clients ("MCCIFs") to super-clients, and super-clients to the hardware modules. Additionally, the MSS implements a system of mapping modules to "software groups" (also known as SWGROUP or SWNAME). This allows a single point of control for all clients for that hardware "engine". Often the list of clients in a SWGROUP is the same as the list of clients in the module that implements the hardware engine.

Each SWGROUP has registers associated with it that select an ASID (Address Space Identifier) and determine whether SMMU translation is enabled or disabled for the client. Normally, each client is associated with one SWGROUP, thus has one ASID associated with it. A few clients have multiple SWGROUPs associated with them, which are selectable on a transaction-by-transaction basis via a 1- or 2-bit SWID flag provided by the client.

The mapping of SWGROUP to modules for Tegra K1 devices is listed in the following table.

SWGROUP	Modules	Translatable?	Further Description
AFI	PCIE	Yes	PCIExpress
AVPC	AVP Cache	Yes	ARM7 Audio-Video Processor (AVP)
DC	Display head 0	Yes(1)	Display reads, head 0, with dis2mc_swid=1'b0
DC1	Display head 0, secure client	Yes(1)	Display reads, head 0, with dis2mc_swid=1'b1 (TZ-Display window)
DCB	Display head 1	Yes	Display reads, head 1 (dis2mc_swid=don't care)
GPU	GPU	No(2)	Kepler GPU with a[34]==0

SWGROUP	Modules	Translatable?	Further Description
GPUB	GPU	Yes[2]	Kepler GPU with a[34]==1
HC	Host1x	Yes	Host interface
HDA	HDA	Yes	High-Definition Audio engine
ISP2	ISP2	Yes	Image Signal Processor
ISP2B	ISP2	Yes	Image Signal Processor (second instance)
MPCORE	AXICIF	No	CPU Complex
MPCORELP	AXICIF_LP	No	Shadow CPU Complex (Tegra K1 32-bit devices only)
MSENC	MSENC	Yes	Multi Standard Video Encoder
PPCS	AHBDMA, AHBSLV	No[3]	Clients in the AHB cluster, with ppcs2mc_swid = 2'b00
PPCS1	AHBDMA, AHBSLV	Yes	Same as PPCS, with ppcs2mc_swid = 2'b01
PPCS2	AHBDMA, AHBSLV	Yes[1]	Same as PPCS, with ppcs2mc_swid = 2'b10
PTC	MC (SMMU)	N/A	Misses from SMMU PTC
SATA	SATA	Yes	Serial ATA
SDMMC1A	SDMMC1	Yes	SDMMC controller. Each SDMMC controller has a unique use case and can be run via standard (Windows/Android) or NVIDIA drivers, hence each controller has its own (hardcoded) SWID.
SDMMC2A	SDMMC2	Yes	
SDMMC3A	SDMMC3	Yes	
SDMMC4A	SDMMC4	Yes	
TSEC	TSEC	Yes	Tegra Security coprocessor
VDE	VDE	Yes	Video Decode Engine
VI	VI	Yes	Video Input engine for CSI, VIP
VIC	VIC	Yes	Video Compositor
XUSB_HOST	USB3 Host	Yes	Although the same driver is expected to control both units, each has its own SWNAME to maintain compatibility with the Tegra 4 device.
XUSB_DEV	USB3 device	Yes	

16. The MC defines the ASID mapping on a per software-ID (SWGROUP) basis. To support secure ASID for the SE engine (in the AHB cluster) and TZ-Display, additional SWID signals are added to the interfaces. The additional SWID signal allows the MC to map the client to different SWIDs, providing flexibility to use a secure ASID for the SMMU.

17. Although the GPU has one physical read client and one physical write client, it can specify one of two SWGROUPs (GPU and GPUB) for a given request, depending on the value of address bit a[34]. LTCX maps a[34] to the MCCIF SWID field. SMMU translation is disabled for SWGROUP GPU and enabled for GPUB. This is how the GPU indicates whether SMMU translation is needed. The MC also performs special video protection region checks for the GPUB.

18. To support the CPU's view of 4GB memory, part of the 0-2G memory address space is used by the CPU to address DRAM. The devices which were previously making requests in the 0-2G address range, assuming that they would be always mapped to other memory space by SMMU, now overlap with devices making physical address requests in the 0-2G range. An additional bit ppcs2mc_swid indicates which requests need to remain untranslated and which requests need SMMU translation. This additional bit selects between the two SWIDs (for AHB clients). These two SWIDs map to two separate ASIDs, where one indicates SMMU translation and the other indicates no translation.

16.4.4 Virtual Addressing: Using the SMMU

For a discussion of the virtual-address translation features provided by the SMMU, read Section 16.5.1 on "Virtual Addressing" below.

In Tegra K1 devices, the CPU has its own MMU and issues physical addresses to the MSS. Thus it is not possible to enable SMMU translation for the CPU.

16.4.4.1 SMMU Initialization

1. For each Virtual Address Space (VAS) to be used, set up the page table in the DRAM. See the subsection on “Virtual Addressing” below for details on the page-table formats supported.
2. For each ASID, set up the pointer to the page-table base and the ASID-wide protection bits. Since the PTB pointers are stored in a RAM, this uses an indirect access mechanism involving two register writes:
    a. Write MC_SMMU_PTB_ASID_0 with the ASID to be modified.
    b. Write MC_SMMU_PTB_DATA_0 with the information for this VAS:
        - Page Table Base pointer
        - Whether non-secure accesses are allowed
        - Whether reads are allowed
        - Whether writes are allowed
3. Assign hardware engines (SWNAMES) to the ASIDs and turn on translation for those engines. This involves writes to various MC_SMMU_<SWNAME>_ASID_0 registers.
4. If needed for Hardware Diagnostics purposes, disable or enable translation within a SWNAME on a per-client basis. This involves writes to various bits in MC_SMMU_TRANSLATION_ENABLE_*_0 registers.
5. If desired, secure the ASIDs, which prevents 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.

The following alternate initialization sequence is used when TrustZone is enabled on Win8, since the UEFI (Boot Loader) is secure, whereas the SW memory driver (NVMEM) is not:

1. Hardware initializes the MC_SMMU_TRANSLATION_ENABLE_*_0 register to enable translation. This is done by enabling the bits by default on cold boot (reset) or warm boot (lp0). Note that software depends on this. The software memory driver cannot configure the MC_SMMU_TRANSLATION_ENABLE_*_0 register, since it is TrustZone protected.
2. The UEFI sets up the display for SMMU translation: SMMU_ASID_DC = x, SMMU_ASID_DCB = x.
3. Enable the SMMU: SMMU_ENABLE=ENABLE to MC_SMMU_CONFIG_0.
4. Later the software memory driver sets up the rest of the ASIDs: SMMU_ASID_xxx = y (y is a non-secure ASID).

16.4.4.2 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 invalidates on the modified entries. Note that the granularity of invalidation 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

16.4.4.3 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.

16.4.5 Clock Frequency Management

Software programs the emcclk and mcclk frequencies to provide the required bandwidth and latency while minimizing power.

1. The performance of some devices are highly sensitive to MC clock frequency. Some drivers dynamically specify a minimum emcclk frequency that meets their needs. Drivers make these requests on behalf of the CPU, AVP , HDMI, the various USB controllers, GPU, MPE (MSENC), and ACTMON.
2. ISO drivers specify their ISO bandwidth requirements, which imply a minimum emcclk frequency. The ISO manager (part of IMP) ensures that the bandwidth requested is feasible within system parameters.
3. Thermal throttling dynamically specifies the highest MC clock frequency that should be allowed.
4. The central clock code computes the new EMC/MC frequency as:

   MIN(MAX(frequency_requests_from_1), ISO_MBps_to_MHz(sum(bandwidth_requests_from_2)), frequency_request_from_3)

16.4.6 DVFS Sequences

DVFS (Dynamic Voltage and Frequency Scaling) is a software feature for controlling system power. A key hardware feature for supporting that is changing SDRAM frequency on the fly with minimal impact on system performance. That indicates the following requirements:

- EMC timing parameters and their corresponding MC arbitration timing parameters need to be adjusted to the optimal performance for the new frequency. These parameters can usually be derived from DRAM datasheet and do not need characterization.
- All trimmer values need to be adjusted for the new frequency. These values need to be characterized across PVT (process, voltage, temperature) and frequencies.

- The pad configurations that can be turned off to save power at low frequencies should also change with frequency change, such as pad terminations and DQS pulls.
- All registers involved in frequency change should be shadowed. That includes the reserved register fields that can possible be turned on only in a certain frequency range. Note that any Hardware Diagnostics feature that has power implications should be considered to be shadowed.
- No software interference should be expected from software after software pulls the trigger to change frequency and before the clock change completes: Software runs on the CPU from SDRAM. Once the EMC blocks the requests to SDRAM, the CPU may be frozen until the EMC finishes frequency change and unblocks the requests. Note that the CPU can also possibly run out of the contents in cache during clock change, so we need to consider both scenarios.
- The delay caused by clock change should not cause functional failures or visible artifacts.

That results in a design as follows:

- The CAR module handshakes with the EMC on clock change. The EMC handshakes with the MC on shadow register latching. The whole process is done by hardware and transparent to software.
- A rough estimation of clock change delay is 2-6 µs for LPDDR3.
- Supports two clock change mode: self-refresh clock change mode and power-down clock change mode. Both LPDDR3 and DDR3 support both modes. We recommend clock change power-down mode for LPDDR3 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 LPDDR3 clock change requires writing mode registers after clock change.
  - To support DLL-ON/DLL-OFF mode switching on DDR3, program mode registers before or after clock change according to the DDR3 specification.

This is a summary of the frequency change process:

- Software turns off features like self-refresh or auto-calibration if necessary to minimize their interference with DVFS sequence.
- Software programs MC/EMC shadow registers for the new frequency.
- Software sets up the appropriate threshold registers in the Display and VI/ISP for *_ready_for_latency_event.
- Software programs EMC clock change flow control registers, such as entering self-refresh before clock change and programming SDRAM mode registers after clock change.
- Software programs the MC/EMC clock source register to trigger the clock change.
- The CAR sends clock change request to the EMC (clock divider and/or clock source).
- The EMC waits for the iso_ready_for_latency_event signal to be asserted. If and while this signal is deasserted, the EMC/MC continues normal DRAM operation.
- The EMC stalls all DRAM commands including refreshes.
- The EMC handshakes with the MC to stalls incoming transactions and waits until EMC pipe is flushed
  Within handshaking, EMC may enters power down state if it is in power-down clock change mode.
- The EMC executes pre-clock change sequence such as entering self-refresh and programming mode register to turn off DLL.
- The EMC latches in shadowed registers and requests the MC to latch in shadow registers.
- The EMC acknowledges the CAR to change clock.

- The CAR changes the MC/EMC clock.
- The CAR deasserts the clock change request to the EMC indicating the clock change is done.
- The EMC executes post-clock-change sequence, such as exiting self-refresh and programming SDRAM mode registers.
- The EMC deasserts the clock change acknowledgement to the CAR.
- The EMC unstalls DRAM commands including refresh and resumes incoming MC transactions.
- Software waits 1µs before continuing.

Software programming sequence on the Tegra K1 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 LPDDR3)
    DISABLED: self-refresh clock change mode (for DDR3)
- At each clock change:
  - Disable DSR (dynamic-self-refresh) if it is currently enabled. There are two ways to do this (software can choose which)
    - EMC_DBG.WRITE_MUX = ACTIVE
      Read-modify-write: EMC_CFG. DYN_SELF_REF = DISABLE
      EMC_DBG.WRITE_MUX = ASSEMBLY
    - Read-modify-write: EMC_CFG. DYN_SELF_REF = DISABLE
      EMC_TIMING_CONTROL.TIMING_UPDATE = 1
      poll for EMC_EMC_STATUS.TIMING_UPDATE_STALLED == 0
  - Program shadow registers.
  - 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. Note that EMC register reads still will not work from this point until the clock change is complete: issuing an EMC register read in this duration will hang the system.
  - (optional) Program unshadowed EMC registers that need to change with clock change
  - (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
  - (self-refresh clock change mode only) Program SELF_REF to exit self-refresh
    SELF_REF.DISABLED = DISABLED
    SELF_REF.DEV_SELECTN = depending on DRAM configuration
  - Program DRAM mode registers by MRS/EMRS (DDR3) or MRW (LPDDR3) to change CL/WL/TWR etc.
    On DLL-OFF to DLL-ON switch on DDR3, DLL is turned on with this step.
  - Program UNSTALL_RW_AFTER_CLKCHANGE = 1
    All following EMC register programming can happen with on-going traffic after clock change

- Wait 1 µs to ensure the EMC programming goes through before programming CLK_SOURCE_EMC.
- Program CLK_SOURCE_EMC to change the clock:
  EMC_2X_CLK_SRC: MC/EMC clock source
  USE_PLLM_UD: use low jitter clock
  MC_EMC_SAME_FREQ: drive EMC versus MC clock ratio at 1:1, otherwise 2:1
  EMC_2X_CLK_DIVISOR: EMC clock divider
- Wait 1 µs, in case the next clock change starts before the current one completes, which is possible if the software runs out of cache.

16.4.7 ZQ Calibration Sequence

ZQ calibration is a feature of LPDDR3 and DDR3 to calibrate R on (the output driver) across PVT (process, voltage, temperature), and also ODT values on DDR3. The EMC supports periodically sending ZQ calibration commands to both DRAMs, requiring only initial configuration after booting. The EMC also supports one time calibration commands at booting.

For cold booting, the boot ROM does a ZQ-init on LPDDR3 and a ZQ-long on DDR3, and then enables periodic ZQ-short.

For warm booting, boot ROM does ZQ-long on both LPDDR3 and DDR3, and then enables periodic ZQ-short.

Self-refresh exit:

There are three cases:

- Software-controlled self-refresh as part of LP0/LP1. The device may spend an arbitrarily long time in either of these low-power states, thus could suffer significant drift temperature since the last ZQ calibration. Software must perform ZQ-long at self-refresh exit.
- Hardware-controlled self-refresh after extended period of idle. The duration should not be long (if the duration were long, software should have triggered LP1). Normal, timer-initiated ZQ-short should be sufficient.
- Hardware-controlled self-refresh as part of DVFS. The self-refresh duration deliberately is kept small since the DVFS sequence is as short as possible. There is no need to perform a ZQ calibration within the DVFS sequence itself, since DRAM voltage is not affected by DVFS. Normal, timer-initiated ZQ-short should be sufficient.

One time calibration:

- LPDDR3, Write MC_MRW:
  - MRW_DEV_SELECTN: 2 for device 0, 1 for device 1. Or use 0 for both (not used because software must control the delays between commands in boot ROM).
  - MRW_MA: 10
  - MRW_OP: 0xFF for ZQ-init. 0xAB for ZQ-long, 0x56 for ZQ-short
- DDR3: Write EMC_ZQ_CAL
  - ZQ_CAL_DEV_SELECTN: 2 for device 0, 1 for device 1. Or use 0 for both (not used because software must control the delays between commands in boot ROM)
  - ZQ_CAL_LENGTH:
    SHORT: issue ZQ-short command
    LONG: issue ZQ-long command
  - ZQ_CAL_CMD: 1

Periodic calibration:

- EMC_ZCAL_INTERVAL: Number of microseconds to wait between periodical ZQ commands. Program this register to 0 to disable periodic ZQ calibration
- EMC_ZCAL_WAIT_CNT: Number of clocks to wait after each ZQ command before other commands
- EMC_ZCAL_MRW_CMD:

- ZQ_MRW_DEV_SELECTN
  2 for device 0 only, 1 for device 1 only, 0 for both devices
- ZQ_MRW_MA
  LPDDR3: 10
  DDR3: 0 for ZQ-short, 1 for ZQ-long
- ZQ_MRW_OP
  LPDDR3: 0x56 for ZQ-short; 0xAB for ZQ-long
  DDR3: 0

16.4.8 MRR Sequence

Mode Register Read (MRR) is desirable on LPDDR3 for reading the manufacturer ID to identify DRAM vendors, or reading back from the LPDDR3 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 LPDDR3 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

16.4.9 Auto-Calibration Sequence

Auto-calibration is for periodically calibrating the output driver of Tegra K1 pads. The EMC supports periodic auto-calibration without software interference, usually initiated at booting.

To enable auto-calibration:

- Program AUTO_CAL_INTERVAL: the number of microseconds between two auto-calibrations
- Program EMC_AUTO_CAL_CONFIG:
  AUTO_CAL_START = 1
  AUTO_CAL_ENABLE = 1
  AUTO_CAL_OVERRIDE: 1 to enable override, 0 to use normal auto-calibration
  AUTO_CAL_PD_OFFSET: override value
  AUTO_CAL_PU_OFFSET: override value
  AUTO_CAL_E_CAL_UPDATE: number of EMC clocks E_CAL_UPDATE is asserted
  AUTO_CAL_STEP: number of microseconds between each calibration step

To disable auto-calibration:

- Program AUTO_CAL_INTERVAL to 0

To re-enable auto-calibration after disabling it:

- Program AUTO_CAL_INTERVAL to the new value

16.4.10 LP0/LP1 Entry/Exit

LP0 and LP1 are both power saving modes with DRAM in self-refresh mode. The only difference is that the MC and the EMC are ON during LP1 but powered OFF during LP0.

LP0 and LP1 entry share the same sequence:

- Disable DSR (Dynamic self-refresh) by read-modify-write EMC_CFG. DYN_SELF_REF = DISABLED then program EMC_TIMING_CONTROL.TIMING_UPDATE = 1 to latch in the shadowed register EMC_CFG
- Wait 5 µs for worst case DSR exit time
- Program EMC_REQ_CTRL to stall all transactions
  STALL_ALL_WRITES = 1
  STALL_ALL_READS = 1
- Wait until the EMC pipe is flushed
  poll EMC_EMC_STATUS.NO_OUTSTANDING_TRANSACTIONS == 1
- Enter self-refresh
  EMC_SELF_REF.SELF_REF_CMD = 1
  EMC_SELF_REF.SREF_DEV_SELECTN: 2 for device/rank 0 only, 1 for device/rank 1 only, 0 for both
- (optional) Wait until the device/rank is in self-refresh
  poll EMC_EMC_STATUS.DRAM_IN_SELF_REFRESH ==
  1: dev0/rank0 is in self-refresh; 2: dev1/rank1 is in self-refresh; 3: both devices/ranks are in self-refresh

LP0-exit is also called warm boot. Refer to the “Warm Boot” subsection.

LP1-exit:

- Exit self-refresh
  EMC_SELF_REF.SELF_REF_CMD = 0
  EMC_SELF_REF.SREF_DEV_SELECTN = 2 for device/rank 0 only, 1 for device/rank 1 only, 0 for both
- poll for self-refresh completes
  poll for EMC_EMC_STATUS.DRAM_IN_SELF_REFRESH == 0
- Program EMC_REQ_CTRL to unstall all transactions
  STALL_ALL_WRITES = 0
  STALL_ALL_READS = 0
- (optional) if it is desired to re-enable DSR (possibly turned off by LP1 entry sequence):Read-modify-write EMC_CFG.
  DYN_SELF_REF = ENABLED
  EMC_TIMING_CONTROL.TIMING_UPDATE = 1 to latch in shadowed EMC_CFG

16.4.11 Performance Tuning

Note: The capitalization in the below sections refers to the actual register name of the associated setting.

16.4.11.1 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

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.

16.4.12 Errors (Interrupts)

Both the MC and the EMC implement a handful of interrupts to inform the software of completed events or error conditions encountered by the hardware. Both the MC and the EMC have implemented the same style of interrupt interface, so this section will use the MC as the example.

The interrupt interface for MC consists of two registers and a hardware output wire. The output signal is wired to the central interrupt handling module on Tegra K1 devices; from there it can be routed to any of the CPU complexes. The first register, INTSTATUS, contains the interrupt vector for MC. Each bit in this register is set when the hardware detects an interrupt. Writing a 1 to the interrupt vector bit will clear the associated interrupt. The second register is INTMASK, each bit in this register corresponds to a vector bit in INTSTATUS. If the MASK bit for an interrupt is *clear (MASKED)*, the corresponding interrupt will *not* forward the interrupt to the central interrupt handling module. If any UNMASKED interrupt vector bits are set, the MC will assert the interrupt to the central interrupt handling module.

16.4.12.1 EMC Interrupts

The EMC generates interrupts on the following conditions:

- DLL alarm: From the EMC Digital DLL when the output delay code reaches the maximum value.
- Attempt to issue a command to a device that is in self-refresh.
- Read data from the MRR is available (to prevent software from polling for data)
- DRAM clock change sequence complete
- Refresh overflow

16.4.12.2 MC Interrupts

The MC contains two classes of interrupts: address decode errors and performance warnings.

When an address decode error occurs, the offending address is captured in the MC_ERR_ADDR register, and information about the error is captured in one or more MC_ERR_*STATUS registers. The captured information assists developers in debugging the error.

A single request can trigger multiple errors. There are multiple error status registers (MC_ERR_*STATUS) that capture the status of different types of errors, as listed below:

- MC_ERR_STATUS – Multiple violations. When more than one of the following violations occurs, the highest-priority violation is reported (listed below in descending priority order):
  - ERR_TYPE = SECURITY_TRUSTZONE – TrustZone carveout violations
  - ERR_TYPE = INVALID_SMMU_PAGE – Any SMMU translation violation

    Decode error on PDE/PTE (reserved bits non-zero)

    Request violates RW, nonsecure PTB/PDE/PTE bit values
  - ERR_TYPE = DECERR_EMEM - DRAM minimum/maximum allowed memory addresses
- MC_ERR_SEC_STATUS - All SEC carveout violations
- MC_ERR_MTS_STATUS - All microcode carveout violations
- MC_ERR_VPR_STATUS - All VPR carveout violations

The capture registers record the following information about the error:

- the Virtual or Physical address of the error (depending on the type of fault)
- which type of fault the captured error corresponds to
- a read/write indicator
- the requestor client ID

- the swap bit sent by the client
- if the error was an INVALID_SMMU_PAGE, then information about the page's protection status is also captured (only in MC_ERR_STATUS):
  - 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.

There is 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 to hit the DEADLOCK_PREVENTION_SLACK_THRESHOLD. In addition to true performance problems, this interrupt may fire in situations such as clock-change where the EMC backpressures pending traffic for long periods of time. This interrupt helps developers identify and debug performance issues and configuration issues.

SMMU-Translated Requests and Physical Aperture Errors

All Virtual Addresses are translated through the SMMU, then run through the Physical Aperture checks. There are a number of confusing scenarios when debugging these interactions including:

- SMMU fault followed by any physical aperture error of any sort: because the SMMU translation occurs first, the data captured by the ERR_ADR and ERR_STATUS registers will always be the SMMU fault information. Both interrupt bits are set.
- Decode error on a successfully translated (i.e., no SMMU fault indicated) client request: the PA checks will throw a decode error; the data captured includes the Physical Address, not the Virtual Address. The client request is squashed. The decode-error interrupt bit is set.
- Decode error on a Page-Table fetch (PDE or PTE): the page-table fetch will throw a decode error; the data captured includes the Physical Address of the PDE or PTE, and the client ID corresponds to the PTC. In addition, the data returned to the PTC is modified to turn off all access permissions, marking the page INVALID. This forces the client request to also throw a SMMU fault when it is translated. Both the decode-error and SMMU-fault interrupt bits are set. The PDE or PTE are then cached with those modified access permissions, so any subsequent requests to that virtual page are also SMMU fault.
- TrustZone security violation on a translated client request: the PA checks throw a security error, the data captured includes the Physical Address, not the Virtual Address. The security violation interrupt bit is set. The client request is squashed. This behavior allows the Secure Aperture to stand as a “last line of defense” against SMMU page-table modification attacks.
- Protected-region security violation on a translated client request: the PA checks throw a security error. The data captured includes the Physical Address, not the Virtual Address. The security violation interrupt bit is set. However, the client request is squashed.
- Video Protection-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.

16.5 Functionality

16.5.1 Addressing

The 34-bit physical address space supported by the MC is 16GB. Clients may support larger (e.g. 40-bit) addresses internally, but address msbs above those connected to the memory controller are assumed to be zero. Clients may use virtual or physical addresses. If a client uses virtual addresses, the addresses are limited to 32-bits (the input width of the SMMU). The SMMU outputs 34-bit physical addresses.

16.5.1.1 Address Map and Aperture

Tegra K1 device's physical address space includes apertures for DRAM, I/O devices, and the AHB redirection region, as defined in the address map (see the section entitled "Address Map"). The range from 0 to 2GB is designated as the MMIO region. All non-DRAM devices (including the AHB redirection region) map (sparsely) within the MMIO region. Address decoders above the MC decode the address ranges for the MMIO devices and intercept requests for the respective devices before the requests reach the MC.

Requests that reach the MC map to one of the following ranges:

- AHB redirection range between IRAM_BOM and IRAM_TOM
- External memory (DRAM) between EMEM_BOM and EMEM_TOM (maximum of 8 GB)
- Not Assigned range (if the address does not fall in either of the two ranges above)

The AHB redirection path allows MC clients that need to access boot media to access IRAM during boot. See the "AHB Redirection Region" section for details on the use and programming of the AHB redirection feature. Note that the AHB redirection range might not map on top of the external memory range, depending on how EMEM_BOM is set.

For accesses that fall within the Not Assigned region, an error is logged, writes are acknowledged and squashed before the WCAM, and reads return all 1s.

Four additional programmable apertures restrict which devices can access defined regions of DRAM:

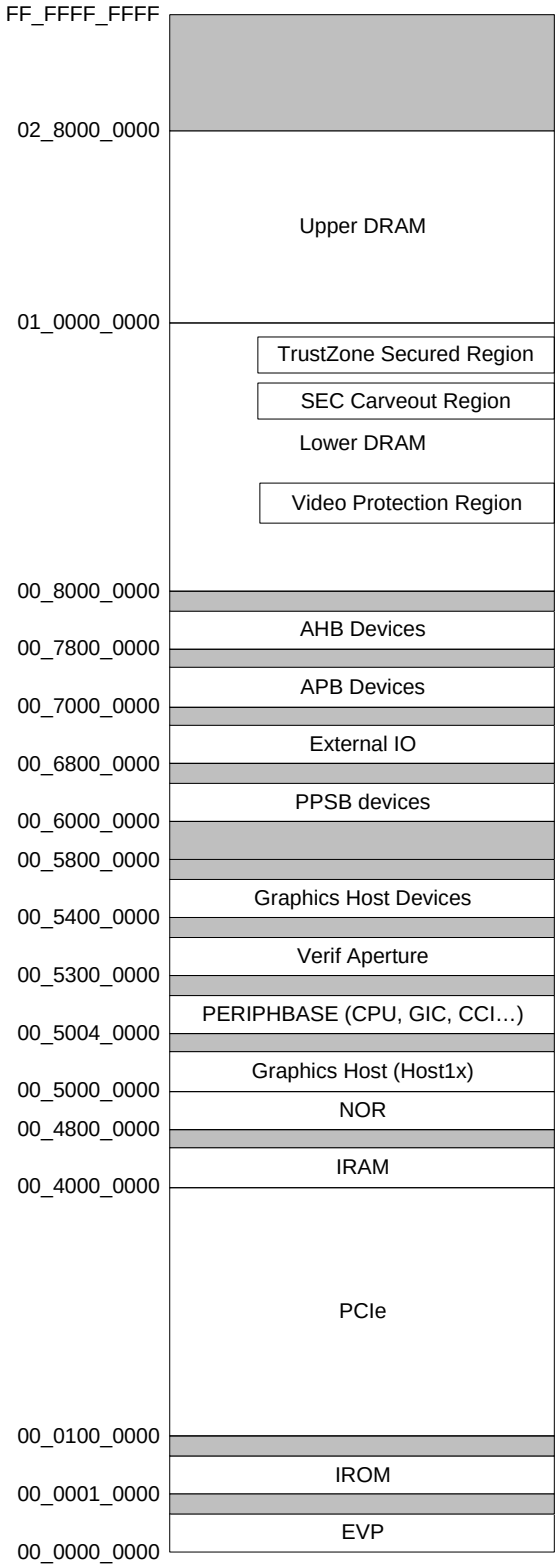
- TrustZone region
- Video Protection region
- Security Coprocessor Secure region
- Microcode Carveout Protection region

Each of these may be placed anywhere in the external memory range and have programmable size. For details on these programmable apertures, see the "TrustZone Security Region", "Video Protection Memory Region", and "Security Coprocessor Secure Region" sections for more information.

In addition, a range of contiguous physical addresses may be designated as "carveout". Virtual memory pages mapping to carveout are required to maintain physical contiguity. Carveout is a software-defined region with no hardware aperture detection.

The following figure shows a possible address map for a Tegra K1 system with 8GB of physical memory. For systems with 32-bit addressing (e.g., Windows 8 systems), all memory must live below the 4 GB boundary. If 4GB of DRAM is desired, the external memory range must be configured to map from 0 to 4GB. DRAM is then mapped in the holes in the sparsely populated MMIO space, providing nearly 4GB of addressable DRAM memory. This is the "Swiss Cheese" approach.

Figure 30: Physical Address Map Example for a System with 8 GB of DRAM



16.5.1.2 Client Types

The Memory Subsystem supports five types of clients:

- **32-bit clients with virtual addressing.** These clients use the SMMU for page-based and ASID-based address protection and translation
- **32-bit clients with physical addressing.** These clients either support native scatter-gather or require regions of contiguous physical addresses. Because of the 32-bit input address width, these clients can only access the low 4GB of physical address space. These clients still have ASID protection for secure regions.
- **34-bit clients with virtual addressing.** These clients use the SMMU for page-based and ASID-based address protection and translation. Because the SMMU only handles 32-bit input addresses, the upper two address bits are ignored when operating in this mode.
- **34-bit clients with physical addressing.** These clients either support native scatter-gather or require regions of contiguous physical addresses.
- **34-bit CPUs.** These clients have their own memory management unit and CANNOT under any circumstance use the SMMU for address protection or translation

For Tegra K1 devices, the following list indicates the clients that support 34-bit addressing:

- CPU Complex
- GPU
- XUSB (USB3)
- AFI (PCIE)
- SATA
- SDMMC
- HDA
- VI2/ISP2 (Virtual address client, so only 32 bits are relevant)
- Display (Virtual address or physical address client. When virtual, only 32 bits are relevant)

16.5.2 Tegra K1 Granularity

The minimum burst length for DDR3 is 8. LPDDR3 supports burst lengths of 8. The Memory Subsystem only supports BL8 DRAM. Because of x64 DDR3, the minimum quantum to transfer is 64B. When using a 32-bit DRAM channel, the EMC will construct two BL8 requests to DRAM to service the 64B request from the MC.

To match this, MC transactions carry 64B of data payload. The 64B can be contiguous or can consist of two non-contiguous 32B “sectors” that lie within the same 512B “gob”, each mapping to a different 32-bit DRAM sub-partition (see the “Memory Sub-Partitions” section). Byte access remains possible but involves a 64B access: write accesses can use byte enables to write a set of bytes (including a null set), and read accesses always return two full 32B sectors with potentially unneeded data (also known as overfetch). To efficiently operate in DDR3 mode, most engines in Tegra K1 devices use 64B memory requests.

The granularity of external access is linked to the burst size programmed for DRAM access. Bursts into each 32-bit DRAM channel always start at a 32B aligned boundary and consist of 8 data beats. For x32 configurations, the two 32B sectors in an MC request correspond to two transfers of 8 data beats.

DDR3 and LPDDR3 support different features to enable less data from being transferred than what the Burst Size indicates. The resulting minimum data transfer is indicated in the following table.

Table 54: LPDDR3 and DDR3 Minimum Data Transfers

LPDDR3	BL8
x32	32B
x64	64B
DDR3	BL8
x32	32B
x32 with Chop	16B
x64	64B

Tegra K1 devices support:

- BL8, LPDDR3 and DDR3
- DDR3 Burst Chop

16.5.3 Virtual Addressing

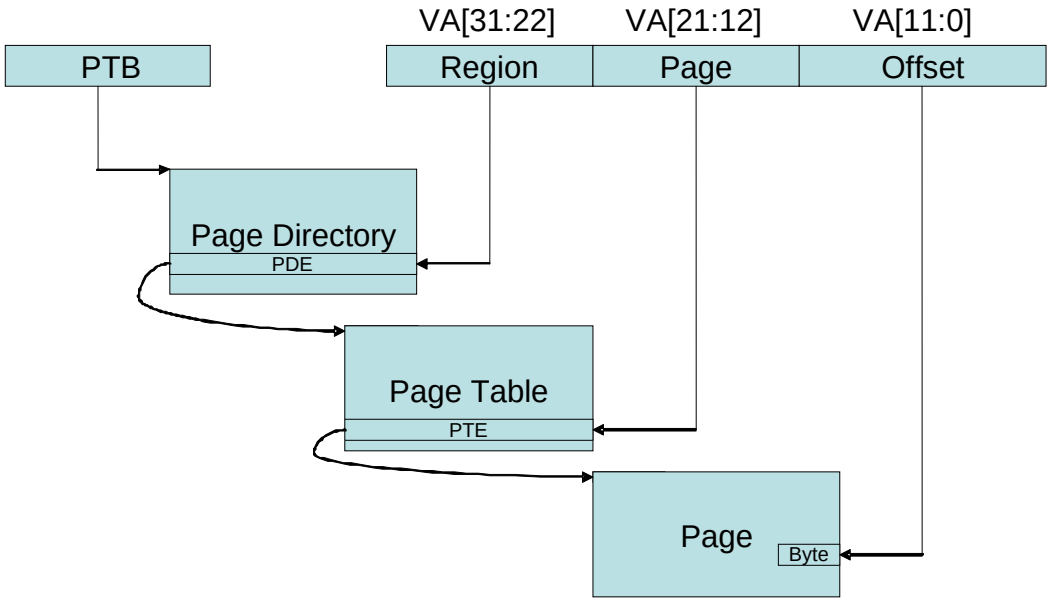
The SMMU is a centralized virtual-to-physical translation for MSS. It maps a 32 virtual address to a 34 bit physical address. If the client address is 40b then bits 39:32 are ignored. It allows the following:

- Memory allocation for hardware pools (surfaces and buffers) could be made from non-contiguous page-sized chunks of memory
- Areas of the hardware memory map could be marked as protected, preventing access from hardware devices
- Hardware memory access can be controlled on a per-device or per-process basis

The virtual-to-physical translation is via a two-level page table that carves up the virtual address as follows:

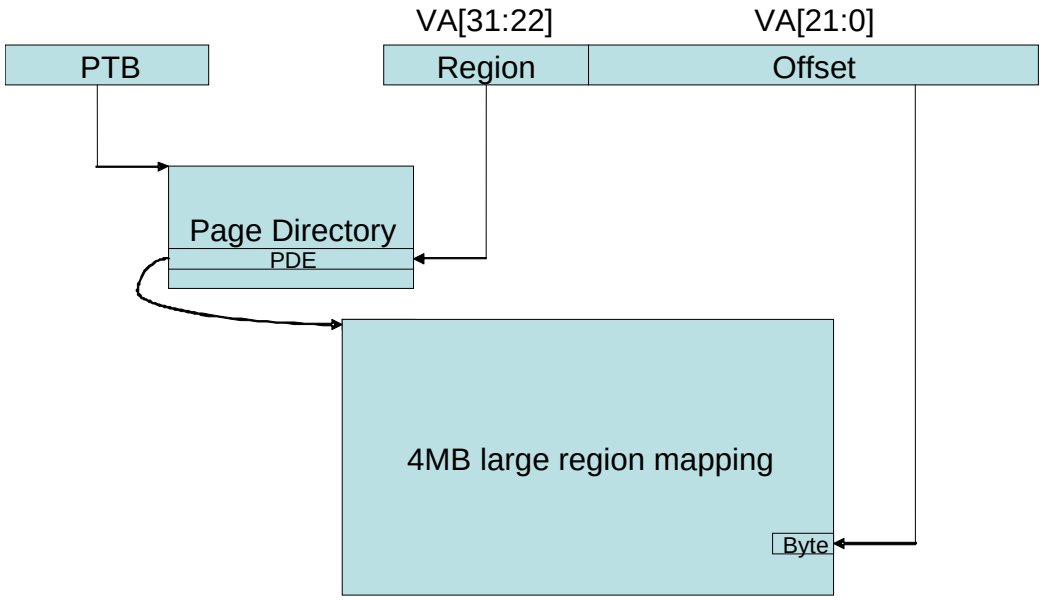
- Bits 31:22 choose a page directory entry (PDE). Each page directory has 1,024 PDEs, each mapping 4MB region.
- Bits 21:12 choose a page table entry (PTE) from a page table. Each page table has 1,024 PDEs, each mapping a 4kB page.
- Bits 11:0 choose a byte within a page

Figure 31: 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 32: Large Region Mapping



The SMMU supports multiple address spaces, each with its own page-table-base register (PTB) and page table. Each address space is tagged with an address space identifier (ASID). This allows for multiple virtual mappings to exist at the same time. Typically, software will assign a different address space to each process that directly uses the hardware drivers, allowing process isolation. The number of ASIDs is 128. Software is responsible for managing the assignments between processes and ASIDs, flushing ASIDs out of the TLB as needed when they are remapped between processes.

ASIDs are assigned based on the SWNAME of the incoming request. An on-chip mapping between SWNAME and the ASID is maintained by the SMMU. For register-based devices, the driver is responsible for updating the mapping between the SWNAME and the ASID when a device in context is swapped between processes.

Untranslated modules are also supported. The SWNAME-to-ASID mapping has a translation-enable bit to allow translation on a per-SWNAME basis. There is also a per-client translation-enable bit for Hardware Diagnostics purposes. This can be used by any client but is typically used by clients that generate 40-bit addresses. In addition, a global “translation enable” bit disables all translation when cleared (cleared is the default out of reset).If translation is disabled, client address bits 33:0 are sent directly to the output of the TU. If the client only provided a 32-bit address, the physical address bits 33:32 will be set to 0.

If translation is enabled, the lower 32 bits of the client address will be used to generate the physical address. Note that this means that client address bits 39:32 will be ignored if a client sends a 40-bit client address and its associated ASID has translation enabled. This is done to minimize the changes to the SMMU. It is expected that 40-bit clients will typically target ASIDs that have translation disabled.

Both the page and directory tables are stored in DRAM. A translation look-aside buffer (TLB) will be used to cache recently-used translations. Translations that miss in the TLB would incur potentially two additional round-trip DRAM delays, but the latency-scheduling features of the Protocol Arbiter will be 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.

To further improve CPU latency, the Tegra K1 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.

16.5.3.1 Page Table Layout

The in-memory layout of a PTE looks like

- 22 bits of physical address (PA[33:12]) 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.

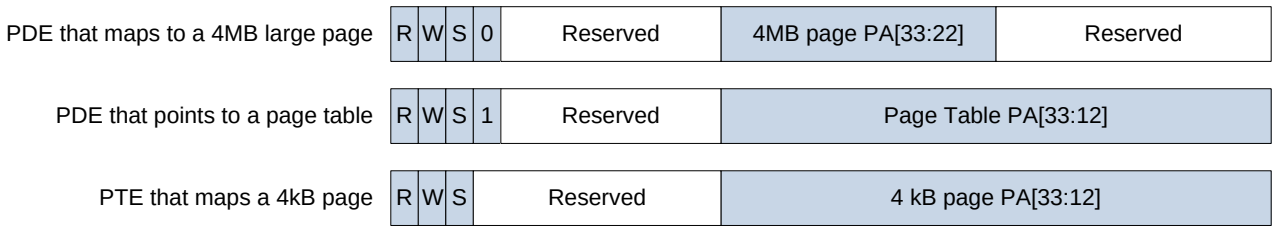
> **Note:** Wherever physical addresses are used in this document, it is assumed that the device can address 8GB of physical memory.

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 22 bits of physical address (PA[33:12]) that point to a page containing the base of the PTE table for this PTE. If the “next level” bit is clear, 12 bits (PA[33:22]) are used 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. Reserved bits should be 0, otherwise they can cause an SMMU fault. Because these are compressed in PTC before they are identified as being PDE or PTE, only the subset of reserved bits that are common to all formats (bits [27:22]) will trigger an SMMU fault.

Figure 33: 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.

16.5.3.2 Page Validity and Protection

There is a need on the software side to distinguish between an invalid mapping (which has an invalid PA field) and a mapping that is valid, but has no access. The following pseudo-code describes the recommended conventions.

```
// page/section/address space permission types
//
// note that INVALID implies the PA field is unused.
// PA is valid for all others, including NO_ACCESS.
enum Permission {
    INVALID               = NvU32(0),
    NO_ACCESS             =                              NONSECURE,
    SECURE_WRITE_ONLY     =             WRITABLE,
    WRITE_ONLY            =             WRITABLE |       NONSECURE,
    SECURE_READ_ONLY      = READABLE,
    READ_ONLY             = READABLE |                   NONSECURE,
    SECURE                = READABLE |  WRITABLE,
    ANY                   = READABLE |  WRITABLE |       NONSECURE
 };
```

For more information on handling SMMU faults, see the subsection on “MC Interrupts” above.

16.5.3.3 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[31:22], or not
- Match VA[31:16] or no VA match

Disabling all matches flushes the entire TLB. Note that the TLB is 16 entries wide, so when you match any particular VA, you flush the entire line, which is why the VA matches are down to 16, rather than 12. 4 MB pages are still matched down to VA[16] since they are stored in the TLB using the 4 KB page format.

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[33:4]) to the PTC flush command. Each flush flushes 16 aligned PTEs or PDEs (one DRAM atom’s worth) out of the cache.

A page table update requires three writes – one to write the PTE to memory, one to flush the stale PTE out of the PTC, and a third to flush the stale VA mapping out of the TLB. However, since a single PTC flush flushes 16 PTEs or PDEs and a single TLB flush can flush the entire TLB, software can reduce the flush traffic by merging flushes from multiple updates.

16.5.4 DRAM Device/Rank, Bank and Page Mapping

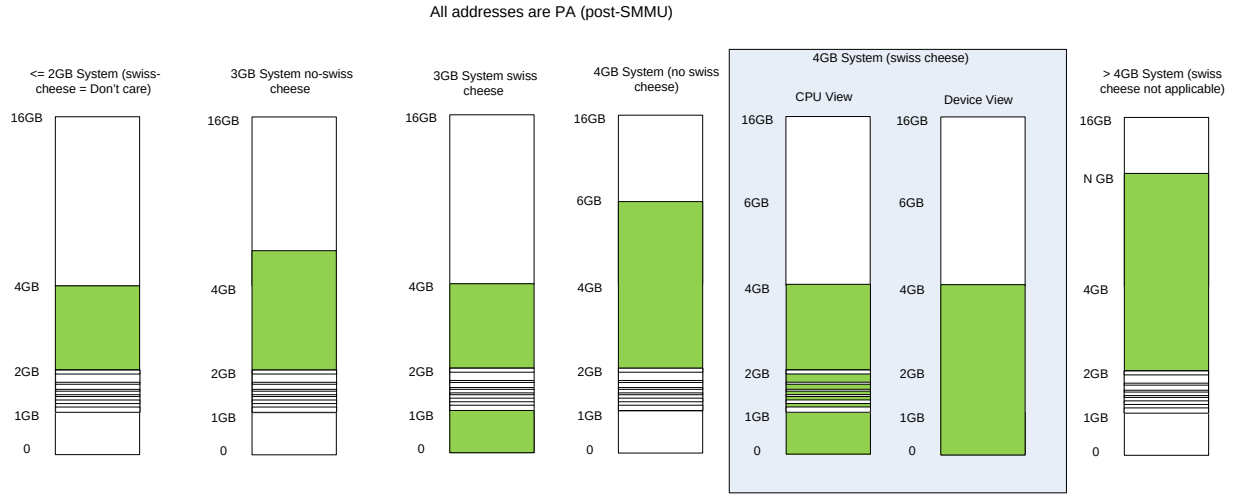
Tegra K1 devices support two modes of addressing; normal addressing mode and swiss-cheese mode. The EMEM_BOM field in the MC_EMEM_CFG_0 register supports the two addressing modes.

EMEM_BOM == 1'b0 : EMEM Base is 2GB // Normal AMAP

EMEM_BOM == 1'b1 : EMEM Base is 0 // Swiss-cheese mode, EAMAP

The various address ranges for DRAM in both configurations are shown the figure below.

Figure 34: Physical DRAM Address Space for Configurations



In case of any errors, MC logs the address of the violating request in the MC_ERR_STATUS_0.ERR_ADR_HI and MC_ERR_ADR_0 registers. In case this error address is due to decerr, the logged address is the original Physical Address.

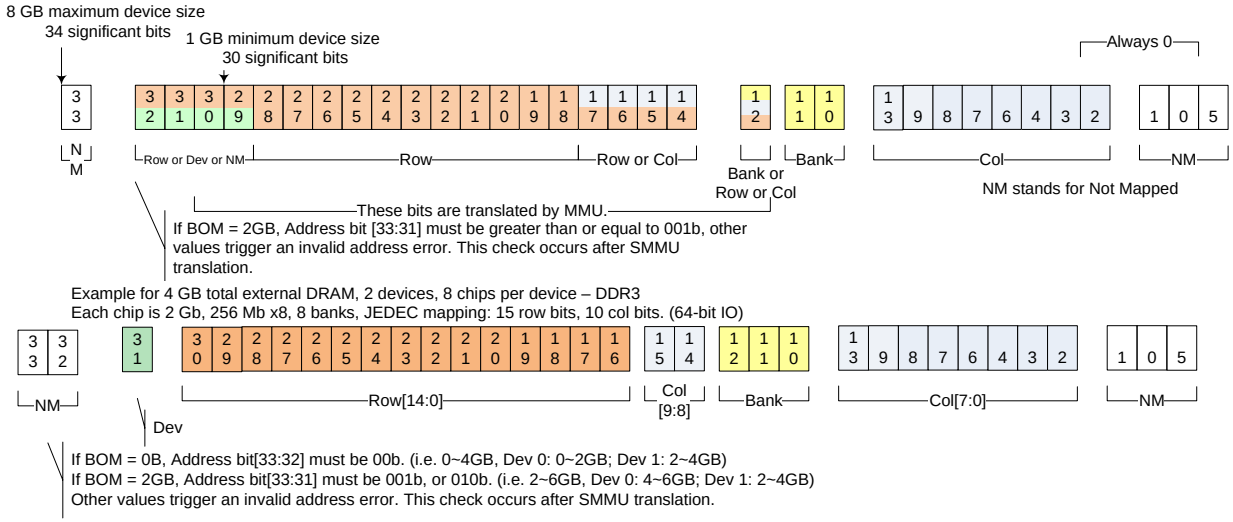
The EMC supports multiple devices with data widths of 8, 16, or 32 bits[2], all identical to form a single rank of either 32 or 64 bits . Up to two ranks are supported.

The bit mapping between the internal linear address and the device/rank, row, column and bank bits is performed as follows:

- The 2 LSBs of the linear address are ignored as the address granularity of the DRAM is 32 bits/4 bytes
- In 64-bit systems, bit 5 of the linear address determines the 32-bit subpartition (with hashing of MSBs)
- In 64-bit systems, bits[9:6,4:2] of the linear address are mapped as column bits [6:0]
- In 32-bit systems, bits[9:2] of the linear address are mapped as column bits [7:0]
- Bits [11:10] are bank bits, possibly hashed.
- Bit [12] is a bank bit if the device has more than two bank bits, possibly hashed.
- 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/rank bit if device/rank bits are needed

[2] A x32 device may be a single x32 chip or two x16 chips in parallel.

Figure 35: Address Mapping Example



The number of sub-partition, bank, column, row, and device bits is limited by the number of address pins available:

- Sub-partition width: 1 or 0
- Bank width: 2 or 3
- Column width: 9 to 12
- Row width: 12 to 16
- Logical Devices/Ranks (also known as chip selects): 1 or 2

When two logical devices/ranks are used, the total memory mapped by the second device/rank must be less than or equal to the first device/rank. The second device/rank also may have a different row, bank, column mapping from the first device.

16.5.5 DRAM Power Modes

The following table maps between the system power modes and DRAM power modes.

Table 55: System Power Modes and DRAM Power Modes

	VDD_CORE	VDD_CPU	MC/EMC Power	PLL	MC/EMC Clock	DRAM State
LP0 (deep sleep)	off	off	off	off	off	self-refresh
LP1 (suspend)	on	off	on	off	off	self-refresh
LP2 (idle)	on	off	on	on	on	active

This subsection briefly describes the Tegra K1 system power modes. Refer to the PMC section for more details.

- LP0: The goal is ultra-long standby time in a cell phone. In this mode, major power rails are off and I/Os are held constant. DRAM is commanded by the AVP to enter self-refresh. The MC and EMC are powered off. Exiting LP0 is called warm boot, in comparison to cold boot which restarts the system. On warm boot, the MC and EMC are re-initialized from the PMC scratch registers and then DRAM is brought out of self-refresh. The LP0 entry code is executed from IRAM and the LP0 exit code is executed from boot ROM, both by the AVP.
- LP1: The goal is to suspend the CPU while allowing the hardware path to be externally accessible. All PLLs are turned off, and therefore MC/EMC/DRAM clocks are frozen. The DRAM is commanded by the AVP to enter self-refresh (it shares the same code with LP0 entry). Because the MC and EMC keep power during LP1, they do not

need re-initialization. To exit LP1, simply enable the clock and exit self-refresh. Both LP1 entry and exit code are executed by the AVP from IRAM.

- LP2: The goal is to power off the CPU while still appearing functional to users. The DRAM can still be accessed by the display, audio, video, and other system modules, although no access comes from the CPU. The LP2 entry/exit sequences do not involve the MC/EMC/DRAM.

For pads and DRAM, LP0 and LP1 are in the same state, and LP2 and normal running mode are in the same state.

16.5.6 Bank Hashing

To achieve a good bank interleave, Tegra K1 devices perform a hash of address MSBs into the bank bits.

16.5.7 AHB Redirection Region

During boot, USB3 and flash media (SDMMC/SATA) devices need access to IRAM. Because these clients connect to the MC and do not have a direct path to the IRAM, The MC implements AHB redirection during boot to allow path to IRAM. In this mode, accesses to a programmed memory address aperture are directed to the AHB bus, allowing access to the IRAM.

The AHB aperture is defined by the IRAM_BOM and IRAM_TOM registers, which are initialized to disable this aperture. If the fuse indicates that the boot media is present in one of these clients, that Boot ROM will initialize this aperture to 0x40000000 and 0x4003f000 (1GB, 1GB+IRAM_SIZE), respectively. Once bootup is complete, the boot ROM programs the IRAM_BOM to 0xffffffff and IRAM_TOM to 0x00000000, thus disabling access to IRAM. DRAM is then potentially accessible in this address range. These aperture register also have an access_control/lock bit. After disabling the aperture, the access_control register should be programmed to lock the registers.

AHB routes the requests targeted to IRAM appropriately and routes all non-DRAM aperture requests (as defined by the swiss-cheese and related configuration) back to the MC as the default DRAM region. Care must be taken to program the IRAM aperture register to prevent a request from being stuck in a loop.

16.5.8 TrustZone Security Region

The MC can secure a region in the physical address aperture from insecure software sources. The region can be defined on 1MB boundaries in the external memory region, in multiples of 1 MB. If a non-secure request is made to the secured region or if a secure request from the CPU is made outside of the secured region (after this mode is enabled), the MC generates an interrupt (if enabled) and logs the details of the request (address, client ID). After the details are logged, write requests are dropped and read requests are forced to return all 1's, thus protecting the secured region from corruption by the insecure source and avoiding aliasing in the CPU cache.

Only clients configured for security can make secure requests.

After the TZ aperture is established, the CPU_STRICT_TZ_APERTURE_CHECK bit in the MC_TZ_SECURITY_CTRL_0 register is set in the MC, which forces correct behavior as follows:

- Secure requests from the CPU can only access TZ secure memory space and not access the memory outside TZ secure memory space.
- Secure requests from other clients may access the TZ secure memory space and also may access memory outside TZ secure memory space.
- Non-secure requests from any client can only access memory outside the TZ secure memory space.

The register MC_TZ_SECURITY_CTRL_0.CPU_STRICT_TZ_APERTURE_CHECK itself is sticky and cannot be changed once set after cold boot.

16.5.9 Video Protection Memory Region

This protected memory aperture prevents the CPU (and other unauthorized clients) from accessing data within the region. The MC secures a region in memory from access and modifications by engines that are not part of the video decode and display process.

This aperture is defined by the VIDEO_PROTECT_BOM and VIDEO_PROTECT_SIZE_MB (reset value 0x0) registers. The value 0 for the size register disables this aperture.

In addition to protecting the region in memory, the registers are allowed only secured access by the Boot Loader only. On cold boot when this register is initially programmed, it is also written to the secure PMC-scratch region. On LP0 resume, the register value is restored from this region. This eliminates need for saving this register content on LP0 entry. An additional lock bit locks down the access to this register. On reset the lock bit is cleared. When the lock bit is cleared, the writes are allowed to the VPR aperture registers. When the VPR registers are initialized, the lock bit is set. When the lock bit is set, all writes to the VPR aperture registers are ignored. This lock bit is used for cold boot as well for LP0 exit.

For systems that do not implement VPR, the Boot Loader must still set the lock bit to indicate that programming is completed.

16.5.10 Security Coprocessor Secure Region

The Security Coprocessor (SEC) engine needs a secured memory space meant exclusively for use by this engine. This memory region, SECR (SEC region) should be accessed only by the SEC engine; no other clients are allowed to read or write to this memory region. This memory region is defined by physical memory range.

The SEC engine (also referred to as TSEC) uses this carveout region as an extension to its I-cache and D-cache. Without this carveout, the TSEC has to encrypt the data before writing to normal DRAM.

The SECR_BOM and SECR_SIZE_MB registers (reset value 0x0) define the 1MB aligned base and size register for this region. In addition to protecting this memory region, these aperture definition registers themselves need to be protected and are allowed to be modified only by the Boot Loader (during cold boot) and the Boot ROM (during LP0 resume) and should be implemented similar to the VPR aperture registers. An additional sticky/secure bit controls this register. On reset this sticky/secure bit is cleared, thus allowing write access to the SECR aperture registers. When the Boot Loader initializes the aperture registers, it also sets this sticky/secure register bit. When this sticky/secure bit is set, all subsequent writes to this register are ignored. On LP0 resume, reset clears this sticky bit; it is set again when the aperture register is restored from the PMC secure scratch space. To avoid saving this aperture register on LP0 entry, the Boot Loader (during cold boot) should store this aperture definition to the PMC secure scratch space.

The size for this carveout region is 1MB.

Any access to this memory aperture by any other client (other than SEC) is ignored (read returns all 1s as the response and writes are dropped) and an error is logged.

16.5.11 Microcode Carveout Protection Region (Tegra K1 64-Bit Only)

Microcode carveout is a protected region of physical memory required by the CPU architecture. Only native CPU accesses from the CCPlex client are allowed to read or write locations in microcode carveout.

The protected region is defined to start at MTS_CARVEOUT_BOM and ends at MTS_CARVEOUT_SIZE. It can start and be extended in multiples of 1MB . The only clients allowed to access the protected region are the CPU read and write clients:

- MPCORER
- MPCOREW
- MPCORELPR
- MPCORELPW

All other clients attempting to access this region will cause the MC to issue an interrupt and log the details of the request (address, client ID). After the details are logged, write requests are dropped and read requests are forced to return all 1s, thus protecting the region from corruption by the insecure source.

The registers defining the aperture also need to be protected. As with the VPR aperture registers, an additional sticky/secure bit controls these registers. On reset this sticky/secure bit is cleared, allowing write access to these registers. When this sticky/secure bit is set, all subsequent writes to these registers are ignored.

The Microcode Carveout Protection Region should be set up and locked at or soon after the time of DRAM configuration, before or during the micro-architectural cold boot process, before any accesses occur to the protected region by the CPU cores. In order to minimize the boot impact, the CPU will begin using the carveout DRAM in parallel with Tegra firmware reading the Boot Loader binary. When entering LP0, microcode will store information sufficient to reconstruct the Microcode Carveout aperture registers into the PMC secure scratch space. On LP0 exit, reset clears the sticky bit; it is set again when the Boot ROM initiates the LP0 DMCE exit sequence to restore the aperture registers from the PMC secure scratch space.

16.6 Memory Tiling

Tegra K1 supports the memory tiling formats described below.

Table 56: Client Shared Tiling Formats

Format	Status	Used for	Description
Pitch Linear	Supported by some clients	Software-compatibility, clients that linearly access memory	Simple array ordering
Block linear 16Bx2 with Tegra K1 sector ordering	Supported by all clients	Kepler color surfaces, clients that access memory in blocks or vertically	Square DRAM page arrangement (blocks) with folded atoms
Legacy (Prior to Tegra 2 devices)	Not Supported		

The parameters associated with each tiling format are listed below:

Table 57: Surface Descriptor Parameters

Parameter	Used by	Description	Values	Minimum Alignment
Layout	All	Surface layout	PITCH, BLOCKLINEAR	N/A
Base Address	Pitch	Starting address of surface, in bytes	32-bit virtual address (40-bit virtual address within the GPU) -or- 34-bit physical address	64B (general) 128B (GPU) 256B (MSENC, VIC)
Pitch	Pitch	Line-to-line stride in bytes	Large enough for at least 4,096 pixels in bytes (some clients may support larger values)	64B (general) 128B (GPU) 256B (MSENC) up to 256B (VIC)[3]
Base Address	Block Linear	Starting address of surface, in bytes	32-bit virtual address (40-bit virtual address within the GPU) -or- 34-bit physical address	512B (one gob)

[3] The pitch alignment restrictions for VIC depend on its cache line shape: 256Bx1: 256B, 128Bx2: 128B, otherwise: 64B.

Parameter	Used by	Description	Values	Minimum Alignment
Block Width	Block Linear	Width of block in gobs	ONE_GOB (this is the only setting supported on 32-bit Tegra K1)	N/A
Block Height	Block Linear	Height of block in gobs	[ONE_GOB, THIRTYTWO_GOBS] (SIXTEEN_GOBS is the normal setting, but ONE_GOB may be needed for display surfaces outside carveout	N/A
Block Depth	Block Linear	Depth of block in gobs	[ONE_GOB, THIRTYTWO_GOBS] (values greater than ONE_GOB are currently only supported by the GPU)	N/A
Width	Block Linear	Width of surface in samples	[1, 4096] samples (some clients may support larger values)	1 sample

(x,y) to linear address translation is the responsibility of clients. The MCCIF only supports linear address inputs. Contact your NVIDIA representative for access to libraries with translation code to simplify implementation.

16.6.1 Block Linear

The block linear formats map batches of contiguous addresses into rectangular *blocks*, and tile these blocks linearly. Blocking serves two purposes: 1) it maps a DRAM page into a rectangular region, rather than into a one-pixel-tall strip, to capture locality in a graphics geometry stream or other 2D-oriented client, such as an MPEG encoder or decoder, and 2) it makes it possible to predictably interleave banks in x and y, minimizing bank conflicts over a large region.

Blocks themselves are arranged from *GOB*s (*Groups of Bytes*). A Kepler GOB is 512 bytes arranged as 64x8 bytes. GOBs are stacked vertically to form a block. The number of GOBs stacked vertically in a block is controlled by an additional surface parameter called the *block height*. The recommended block height for most buffers in Tegra K1 devices is 16 GOBs (128 lines). This supports 8x8 bank interleaving. For buffers for which linear display access is more important than access by GPU, VIC, or other block-oriented engine, block height can be set to 1 GOB, providing more locality for the display client.

Blocks are arranged horizontally into a *Row of Blocks (ROBs)* to fill out the surface to the width value (or slightly beyond, if the width is not a multiple of the 64B block width). Upper address bit swizzling in the bank swizzling algorithm generally makes it unnecessary to pad the surface width further.

16.6.2 Sub-Partitions

The 64-bit DRAM channel can be divided into two sub-partitions with DQ[31:0] forming sub-partition A and DQ[63:32] forming sub-partition B.

Each 64B atom is divided into 2 sectors of 32B. One sector contains bytes 0 – 31 while the other contains bytes 32 – 63.

16.6.3 Software Guidelines

Software should allocate surfaces respecting the minimum alignments specified in Table 57. This section gives additional alignment restrictions for certain clients and advice on allocating surfaces for best performance.

Certain clients, such as MSENC, VDE, and VIC use 32-bit registers (left-shifted 8) to specify 40-bit base addresses of a buffer. Base addresses for these clients must be aligned to a minimum of 256B.

Certain clients, such as MSENC and VIC, have a fixed cache line size and read all data within a cache line, even if it is beyond the nominal height and width of the surface. For such clients, surface width and height need to be padded to at least the next multiple of the cache line size so the unit does not read beyond the end of the buffer.

Surface Type

- **Block linear** format generally provides best performance and should be chosen by default.
- **Pitch format** may be preferable if producer and consumer clients perform linear accesses and both clients support pitch.
- **Private formats** are used by a few units, such as MSENC and VDE. For private formats, the client-required allocation and padding rules must be followed.

Block Linear

- **Alignment**. Align surface start address as follows:
  - 512B is the minimum alignment for correct behavior. Performance will be poor (no zero-bandwidth clears, poor page locality).
  - 1KB alignment is better. Zero bandwidth clears will work. Bank interleave will be suboptimal.
  - 8KB alignment is best. All compression features work. Bank interleave assumes 8KB alignment. Use this setting for large buffers.
- **Block height** should normally be programmed to SIXTEEN_GOBS (128 lines). For certain surfaces for which display is the primary client, block heights of ONE_GOB or TWO_GOBS are recommended. Surfaces to be accessed by the GPU texture unit may require a smaller block height under certain conditions.[4]
- **Width** is typically prescribed by the application. Hardware allocates a block linear surface to be an integral number of blocks wide, so its allocated width is width*Bpp rounded up to the next block boundary (64B), which is sufficient padding in almost all circumstances.[5]
- **Height padding**. Surfaces should be an integral number of blocks tall. If a client such as MSENC or VIC has a cache line that is taller than the block height, additional row(s) of blocks may be required for padding.

Pitch

- **Alignment**. Align surface start address to a minimum of:
  - 128B if surface is to be accessed by the GPU
  - 256B if surface is to be accessed by MSENC, VDE, or VIC (due to the 8-bit left-shift issue)
  - 64B otherwise
- **Pitch**. Surface pitch must be a multiple of:
  - 128B bytes if surface is to accessed by GPU
  - 128B if surface is to be read by VIC with 128x2 cache line shape
  - 256B if surface is to be read by VIC with 256Bx1 cache line shape
  - 64B otherwise.
- **Height padding**. For clients, such as MSENC and VIC, additional lines of padding might be required beyond the nominal height of the surface. For example, when the VIC is reading a source buffer with an internal cache line shape of 32x8, the source buffer must be a multiple of 8 lines tall.

[4] For buffers that are to be accessed using the GPU texture unit, the BlockHeight parameter must match that used by the texture unit. In particular, note that if a texture surface is smaller than the specified size in blocks, texture uses a shrunken blockHeight instead of the nominal parameter specified in the Texture Header.

[5] If VIC cacheline shapes of 128x2 or 256x1 are used (generally not recommended for block linear), width must be padded up to a multiple of the cache line width. If this is not possible, a narrower cache line shape must be used

16.7 Memory Controller Registers

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

16.7.1 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 the top of Section 16.7.1

USAGE NOTE: Many MC registers should be programmed by the Boot Loader as part of the warm boot (also known as "wake from LP0") and cold boot (also known as "power up") sequences. Suggested actions for the Boot ROM/Boot Loader are noted under "Boot requirements:".

USAGE NOTE: Some MC registers may only be accessed by TrustZone-secured register transactions. See the appropriate ARM Architecture Reference Manual for information on how to handle security. Such registers are marked by a comment:

“Access to this register is restricted to TrustZone-secured requestors.”

USAGE NOTE: The MC register fields to be stored in PMC scratch registers for warm boot must have "[PMC]" in their comments. After adding and removing such PMC flag, the tool warmboot_code_gen must be rerun to generate new boot ROM code.

USAGE NOTE: When writing to a register, any bits that are not intended to be modified should be rewritten with their existing values.

USAGE NOTE: Unspecified bits may not appear in tables and should be written with their Reset values.

16.7.1.1 MC_INTSTATUS_0

Interrupt Status

This register contains trigger bit fields. Bit fields not shown here are reserved.

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xx000000x0xxxxxx)

Bit	Reset	Description
16	CLEAR	DECERR_MTS_INT: This interrupt is for access violation on MTS carveout region in the MC. If a client other than CPU read or write is trying to access the region. 0 = CLEAR 1 = SET
13	CLEAR	SECERR_SEC_INT: The request violated the SEC Carveout requirements - Only TSEC clients may access the SEC carveout. 0 = CLEAR

Bit	Reset	Description
		1 = SET
12	CLEAR	DECERR_VPR_INT: The request violated the VPR requirements - in case of a read, the request address fell in the VPR range, but the VPR attribute of the request was not set. In case of a write, the request had the VPR attribute set in the request, but the request address did not fall in the VPR range. 0 = CLEAR 1 = SET
11	CLEAR	INVALID_APB_ASID_UPDATE_INT: ASID Update through APB interface resulted in an error. APB interface tried to update secure ASID through Non-secure interface. 0 = CLEAR 1 = SET
10	CLEAR	INVALID_SMMU_PAGE_INT: Address translation error: An access was attempted to an invalid page or a protected page. See ERR_STATUS and ERR_ADR for more information. 0 = CLEAR 1 = SET
9	CLEAR	ARBITRATION_EMEM_INT: Warning that a pending request has reached the deadlock-prevention slack threshold. This indicates that the MC could not meet the programmed performance goals. This interrupt is for use with assisting debug of performance-related issues. 0 = CLEAR 1 = SET
8	CLEAR	SECURITY_VIOLATION_INT: Address translation error: A nonsecure access was attempted to a secured region. See ERR_STATUS and ERR_ADR for more information. 0 = CLEAR 1 = SET
6	CLEAR	DECERR_EMEM_INT: Address translation error: EMEM Address Decode Error. See ERR_STATUS and ERR_ADR for more information. 0 = CLEAR 1 = SET

16.7.1.2 MC_INTMASK_0

Interrupt Masks

Boot requirements: This register should be saved to SDRAM registers and restored by the OS during warm boot. Bit fields not shown here are reserved.

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xx000000x0xxxxxx)
Default: 0x00003540

Bit	Reset	SW Default	Description
16	MASKED	_NONE_	DECERR_MTS_INTMASK: 0 = MASKED 1 = UNMASKED
13	MASKED	UNMASKED	SECERR_SEC_INTMASK: Prevents SECERR_SEC_INT from triggering an interrupt to the interrupt controller. 0 = MASKED 1 = UNMASKED
12	MASKED	UNMASKED	DECERR_VPR_INTMASK: Prevents DECERR_VPR_INT from triggering an interrupt to the interrupt controller. 0 = MASKED 1 = UNMASKED
11	MASKED	_NONE_	INVALID_APB_ASID_UPDATE_INTMASK: Prevents INVALID_APB_ASID_UPDATE_INT from triggering an interrupt to the interrupt controller. 0 = MASKED 1 = UNMASKED

Bit	Reset	SW Default	Description
10	MASKED	UNMASKED	INVALID_SMMU_PAGE_INTMASK: Prevents INVALID_SMMU_PAGE_INT from triggering an interrupt to the interrupt controller. 0 = MASKED 1 = UNMASKED
9	MASKED	_NONE_	ARBITRATION_EMEM_INTMASK: Prevents ARBITRATION_EMEM_INT from triggering an interrupt to the interrupt controller. 0 = MASKED 1 = UNMASKED
8	MASKED	UNMASKED	SECURITY_VIOLATION_INTMASK: Prevents SECURITY_VIOLATION_INT from triggering an interrupt to the interrupt controller. 0 = MASKED 1 = UNMASKED
6	MASKED	UNMASKED	DECERR_EMEM_INTMASK: Prevents DECERR_EMEM_INT from triggering an interrupt to the interrupt controller. 0 = MASKED 1 = UNMASKED

16.7.1.3 MC_ERR_STATUS_0

Error Data Capture: Status

If a translation error is detected (for example, address-out-of-bounds, security fault, protection fault), this register records information about this access:

- which fault the captured error corresponds to
- a read/write indicator and
- the request ID (global client ID assigned by the memory controller, see the list below).

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, Page Directory Entry, and Page Table Entries 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 faults (of any type) will not change the status and address registers until the corresponding interrupt is cleared.

The global memory client IDs are defined as follows:

Client	ID	ID (hex)	SWGROUP	INBAND-SWID	Module	Partition	Description
csr_ptcr	0	(0x00)	ptc				Misses from System Memory Management Unit (SMMU) Page Table Cache (PTC)
csr_display0a	1	(0x01)	dc		display	dis	Display reads, window A
csr_display0ab	2	(0x02)	dcb		displayb	disb	Display reads, window A
csr_display0b	3	(0x03)	dc		display	dis	Display reads, window B

Client	ID	ID (hex)	SWGROUP	INBAND-SWID	Module	Partition	Description
csr_display0bb	4	(0x04)	dcb		displayb	disb	Display reads, window B
csr_display0c	5	(0x05)	dc		display	dis	Display reads, window C
csr_display0cb	6	(0x06)	dcb		displayb	disb	Display reads, window C
csr_afir	14	(0x0e)	afi		afi	pcx	PCIE reads
csr_avpcarm7r	15	(0x0f)	avpc		avpc	avp	ARM7 Audio-Video Processor (AVP) reads via the avp_cache
csr_displayhc	16	(0x10)	dc		display	dis	Display reads, cursor
csr_displayhcb	17	(0x11)	dcb		display	disb	Display reads, cursor
csr_hdar	21	(0x15)	hda		hda	apb	High-definition audio (HDA) reads
csr_host1xdmar	22	(0x16)	hc		host1x	host	Host channel data reads
csr_host1xr	23	(0x17)	hc		host1x	host	Host indirect reads
csr_msencsrd	28	(0x1c)	msenc		msenc	mse	
csr_ppcsahbdmar	29	(0x1d)	ppcs		ppcs	ahb	AHB DMA engine reads
csr_ppcsahbslvr	30	(0x1e)	ppcs	ppcs1 ppcs2	ppcs	ahb	AHB bus reads
csr_satar	31	(0x1f)	sata		sata	sax	SATA reads
csr_vdebsevr	34	(0x22)	vde		vde2x	vd	Video Decode (VDE) video bitstream engine reads
csr_vdember	35	(0x23)	vde		vde2x	vd	Video Decode (VDE) macroblock engine reads
csr_vdemcer	36	(0x24)	vde		vde2x	vd	Video Decode (VDE) motion-compensation engine reads
csr_vdetper	37	(0x25)	vde		vde2x	vd	Video Decode (VDE) transform engine reads
csr_mpcorelpr	38	(0x26)	mpcorelp		mpcorelp	stop	Reads from Cortex-A9 Shadow CPU core via the L2 cache
csr_mpcorer	39	(0x27)	mpcore		mpcore	ftop	Reads from Cortex-A9 4 CPU cores via the L2 cache
csw_msencswr	43	(0x2b)	msenc		msenc	mse	
csw_afiw	49	(0x31)	afi		afi	pcx	PCIE writes

Client	ID	ID (hex)	SWGROUP	INBAND-SWID	Module	Partition	Description
csw_avpcarm7w	50	(0x32)	avpc		avpc	avp	Arm7 Audio-Video Processor (AVP) writes via the avp_cache
csw_hdaw	53	(0x35)	hda		hda	apb	High-definition audio (HDA) writes
csw_host1xw	54	(0x36)	hc		host1x	host	Host indirect writes
csw_mpcorelpw	56	(0x38)	mpcorelp		mpcorelp	stop	Writes from Cortex-A9 Shadow CPU core via the L2 cache
csw_mpcorew	57	(0x39)	mpcore		mpcore	ftop	Writes from Cortex-A9 4 CPU cores via the L2 cache
csw_ppcsahbdmaw	59	(0x3b)	ppcs		ppcs	ahb	AHB DMA engine writes
csw_ppcsahbslvw	60	(0x3c)	ppcs	ppcs1 ppcs2	ppcs	ahb	AHB bus writes
csw_sataw	61	(0x3d)	sata		sata	sax	SATA writes
csw_vdebsevw	62	(0x3e)	vde		vde2x	vd	Video Decode (VDE) video bitstream engine writes
csw_vdedbgw	63	(0x3f)	vde		vde2x	vd	Video Decode (VDE) debug writes
csw_vdembew	64	(0x40)	vde		vde2x	vd	Video Decode (VDE) macroblock engine writes
csw_vdetpmw	65	(0x41)	vde		vde2x	vd	Video Decode (VDE) transform engine writes
csr_ispra	68	(0x44)	isp2		isp2	isp	ISP Read client for Crossbar A
csw_ispwa	70	(0x46)	isp2		isp2	isp	ISP Write client for Crossbar A
csw_ispwb	71	(0x47)	isp2		isp2	isp	ISP Write client Crossbar B
csr_xusb_hostr	74	(0x4a)	xusb_host		xusb_host	usbx	XUSB reads
csw_xusb_hostw	75	(0x4b)	xusb_host		xusb_host	usbx	XUSB_HOST writes
csr_xusb_devr	76	(0x4c)	xusb_dev		xusb_dev	usbd	XUSB reads
csw_xusb_devw	77	(0x4d)	xusb_dev		xusb_dev	usbd	XUSB_DEV writes
csr_isprab	78	(0x4e)	isp2b		isp2b	ve2	ISP Read client for Crossbar A; instance b of isp2
csw_ispwab	80	(0x50)	isp2b		isp2b	ve2	ISP Write client for Crossbar A; instance b of isp2

Client	ID	ID (hex)	SWGROUP	INBAND-SWID	Module	Partition	Description
csw_ispwbb	81	(0x51)	isp2b		isp2b	ve2	ISP Write client Crossbar B; instance b of isp2
csr_tsecsrd	84	(0x54)	tsec		tsec	apb	TSEC Memory Return Data Client Description
csw_tsecswr	85	(0x55)	tsec		tsec	apb	TSEC Memory Write Client Description
csr_a9avpscr	86	(0x56)	a9avp		a9avpsc	a9avppc	Reads from Cortex-A9 Shadow CPU core via the L2 cache
csw_a9avpscw	87	(0x57)	a9avp		a9avpsc	a9avppc	Writes from Cortex-A9 Shadow CPU core via the L2 cache
csr_gpusrd	88	(0x58)	gpu	gpub	gpu	gk	3D, ltcx reads
csw_gpuswr	89	(0x59)	gpu	gpub	gpu	gk	3D, ltcx writes
csr_displayt	90	(0x5a)	dc	dc1	display	dis	Display reads, Window T
csr_sdmmcra	96	(0x60)	sdmmc1a		sdmmca	aud	sdmmca memory read client
csr_sdmmcraa	97	(0x61)	sdmmc2a		sdmmcaa	aud	sdmmca memory read client; instance a of sdmmca
csr_sdmmcr	98	(0x62)	sdmmc3a		sdmmc	aud	sdmmc3 memory read client
csr_sdmmcrab	99	(0x63)	sdmmc4a		sdmmcab	sd	sdmmca memory read client; instance b of sdmmca
csw_sdmmcwa	100	(0x64)	sdmmc1a		sdmmca	aud	sdmmca memory write client
csw_sdmmcwaa	101	(0x65)	sdmmc2a		sdmmcaa	aud	sdmmca memory write client; instance a of sdmmca
csw_sdmmcw	102	(0x66)	sdmmc3a		sdmmc	aud	sdmmc3 memory write client
csw_sdmmcwab	103	(0x67)	sdmmc4a		sdmmcab	sd	sdmmca memory write client; instance b of sdmmca
csr_vicsrd	108	(0x6c)	vic		vic	vicpc	
csw_vicswr	109	(0x6d)	vic		vic	vicpc	
csw_viw	114	(0x72)	vi		vi2	ve	VI Write client
csr_displayd	115	(0x73)	dc		display	dis	Display reads, Window D

Offset: 0x8 | Read/Write: RO | Reset: 0xXXXXX0XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:28	X	ERR_TYPE: Type of the corresponding error: 0 = RSVD : Code 0x0 is never used. 2 = DECERR_EMEM : There is no physical DRAM attached for this address based on the value of EMEM_SIZE_MB configuration register. 3 = SECURITY_TRUSTZONE : The non-secure client has attempted to access the Trustzone-secured aperture; or a secure client has attempted to access the Trustzone-secured aperture using a non-secure request. 4 = SECURITY_CARVEOUT : Reserved 6 = INVALID_SMMU_PAGE : The SMMU page-translation flagged an error for that access.
27	X	ERR_INVALID_SMMU_PAGE_READABLE: If INVALID_SMMU_PAGE error, set if page permission was readable (AND of PTB/PDE/PTE). Ignore if no INVALID_SMMU_PAGE error.
26	X	ERR_INVALID_SMMU_PAGE_WRITABLE: If INVALID_SMMU_PAGE error, set if page permission was writable (AND of PTB/PDE/PTE). Ignore if no INVALID_SMMU_PAGE error.
25	X	ERR_INVALID_SMMU_PAGE_NONSECURE: If INVALID_SMMU_PAGE error, set if page permission was non-secure (AND of PTB/PDE/PTE). Ignore if no INVALID_SMMU_PAGE error.
21:20	X	ERR_ADR_HI: Higher address bits of erring address whose lower bits are available in ERR_ADR register
18	X	ERR_SWAP: Swap bit for transaction
17	X	ERR_SECURITY: Set if the transaction was secure. 0 = NONSECURE 1 = SECURE
16	X	ERR_RW: Direction of the access that caused the error. 0 = READ 1 = WRITE
14:12	X	ERR_ADR1: Second subpartition unique address bits
6:0	X	ERR_ID: Client identifier (see above list) of the access that caused the error.

16.7.1.4 MC_ERR_ADR_0

Error Data Capture: Address

If a translation error is detected, this register records the request byte address.

Subsequent faults (of any type) will not change the status and address registers until the corresponding interrupt is cleared.

Note that the decode-error, security-Trustzone, and security-carveout checks occur on the physical address (in other words, after SMMU translation); thus those error types capture the physical address. All other errors capture the virtual address.

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

Bit	Reset	Description
31:0	X	ERR_ADR

16.7.1.5 MC_SMMU_CONFIG_0

Used for interrupt set/clear enums. When disabled, all transactions are untranslated. When enabled, transactions may be translated. See per-client and per-module enables in SMMU_* registers below.

This register can only be accessed by TrustZone-Secured accesses from the CPU.

Access to this register is restricted to TrustZone-secured requestors.

Boot requirements: This register should be saved to SDRAM registers and restored by the OS during warm boot.

SMMU Enable Register

Offset: 0x10 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	DISABLE	SMMU_ENABLE: 0 = DISABLE 1 = ENABLE

16.7.1.6 MC_SMMU_TLB_CONFIG_0

Translation Lookaside Buffer Config Register

Controls usage of the TLB.

Boot requirements: This register should be saved to SDRAM registers and restored by the OS during warm boot.

Offset: 0x14 | Read/Write: R/W | Reset: 0x30000020 (0b0011xxxxxxxxxxxxxxxxxxxxx100000)

Bit	Reset	Description
31	DISABLE	TLB_STATS_ENABLE: Enable TLB Hit and Miss counters 0 = DISABLE 1 = ENABLE
30	DISABLE	TLB_STATS_TEST: Set stats registers to all "1s" 0 = DISABLE 1 = ENABLE
29	ENABLE	TLB_HIT_UNDER_MISS: Allow hits to pass misses in the TLB. This value may not be changed on the fly. Ideally, this should be set before enabling the SMMU. At the very least, the TLB needs to be flushed and traffic through the SMMU stopped before changing this value. 0 = DISABLE 1 = ENABLE
28	ENABLE	TLB_ROUND_ROBIN_ARBITRATION: When enabled, forces round robin (RR) arbitration between TLB hit and miss FIFOs. 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.

16.7.1.7 MC_SMMU_PTC_CONFIG_0

Controls usage of the PTC

Boot requirements: This register should be saved to SDRAM registers and restored by the OS during warm boot.

Page Table Cache Config Register

Offset: 0x18 | Read/Write: R/W | Reset: 0x2800003f (0b001x1000xxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
31	DISABLE	PTC_STATS_ENABLE: Enable PTC Hit and Miss counters 0 = DISABLE 1 = ENABLE
30	DISABLE	PTC_STATS_TEST: Set stats registers to all "1s" 0 = DISABLE 1 = ENABLE
29	ENABLE	PTC_CACHE_ENABLE: Enable the PTC cache. 0 = DISABLE 1 = ENABLE
27:24	0x8	PTC_REQ_LIMIT: Limit outstanding PTC fill requests to the DRAM. Minimum value is 5; otherwise the SMMU can have functional failures.
5:0	0x3f	PTC_INDEX_MAP: XOR pattern for tag generation

16.7.1.8 MC_SMMU_PTB_ASID_0

Page Table Bases (PTBs)

These pointers point to a 4KB-page aligned table of PDEs for each ASID. They also contain initial permission bits that are ANDed with those in the PDE and PTEs to determine the final permissions of accesses through this ASID.

The PTBs are stored in a RAM that is accessed indirectly though the SMMU_PTB_ASID and SMMU_PTB_DATA registers.

Boot requirements: This RAM should be saved to SDRAM and restored by the OS during warm boot.

Page Table Base ASID Register

This register selects which PTB is modified by the SMMU_PTB_DATA register.

Offset: 0x1c | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
6:0	X	CURRENT_ASID: ASID used to address SMMU_PTB register

16.7.1.9 MC_SMMU_PTB_DATA_0

Writes or reads to this register write or read the Page Table Base pointer for the ASID defined in the SMMU_PTB_ASID register.

This register is not TrustZone secure, but only a TrustZone secure access may alter the PTB for a secure ASID.

Secure ASIDs have additional privilege over other non-secure ASIDs - they may fetch page tables from secure memory and they can allow clients translated through them to make secure accesses if programmed to do so.

Attempts to write a secure ASID with a non-secure access will be ignored, and non-secure reads will return garbage.

Page Table Base Data Register

Offset: 0x20 | Read/Write: R/W | Reset: 0xX0XXXXX (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
21:0	X	ASID_PDE_BASE: Pointer to page of PDEs, bits [33:12] for this ASID

16.7.1.10 MC_SMMU_TLB_FLUSH_0

Software can write this register to flush a specific page group, 4MB section, entire ASID or the entire TLB. There is independent control over matching part or all of the VA and/or matching the ASID.

Note: The granularity of VA mapping here is at a page group, which is the number of PTEs that fit in a memory atom.

Translation Lookaside Buffer Flush Register

Offset: 0x30 | Read/Write: R/W | Reset: 0xXX0XXXXX (0bxxxxxxxxxxx0xxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TLB_FLUSH_ASID_MATCH: If enabled, only entries matching TLB_FLUSH_ASID are flushed. 0 = DISABLE 1 = ENABLE
30:24	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[31:15]. VA[14] of this field is ignored due to the TLB line size.
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. By default, the GROUP setting takes effect if this field is programmed to a value of 1. 0 = ALL 2 = SECTION 3 = GROUP

16.7.1.11 MC_SMMU_PTC_FLUSH_0

Software can use this register to flush a specific PTE or PDE from the Page Table Cache (PTC) by writing the atom-aligned physical address of the PTE group. It can also flush the entire PTC via this register.

Note: The granularity of flushing is at the PTE group, which is the number of PTEs that fit in a memory atom.

Page Table Cache Flush Register

Offset: 0x34 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx)

Bit	Reset	Description
31:4	X	PTC_FLUSH_ADR: Physical address of PTE group to match for address flushes, PA[31:4]
3	0x1	PTC_NO_WAIT_IDLE_FLUSH: Set bit to immediately flush without waiting for hit FIFO to go idle. Not required because it does not present an ordering hazard.

Bit	Reset	Description
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

16.7.1.12 MC_EMEM_CFG_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.
- During warm boot, this register may be derived from EMEM_ADR_CFG* register values.

External Memory Aperture Configuration

Offset: 0x50 | Read/Write: R/W | Reset: 0x00002000 (0b0xxxxxxxxxxxxxxxxx10000000000000) | Default: 0x00000000

Bit	Reset	SW Default	Description
31	B2GB	B2GB	EMEM_BOM: Base of DRAM memory in GBs [PMC_SECURE]. B2GB should be used for 34b systems, while B0GB supports up to 4GB using extended mapping mode 1 = B0GB 0 = B2GB
13:0	0x7ff	NONE	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.

16.7.1.13 MC_EMEM_ADR_CFG_0

The EMEM_ADR_CFG* registers are used to specify the DRAM density and geometry parameters. The 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.

Valid settings for EMEM_ADR_CFG_DEV* registers should ensure device address width <= bank width + column width + NV_MC_EMEM_ROW_WIDTH

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

External Memory Address Configuration, System

Offset: 0x54 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	N1	EMEM_NUMDEV: [PMC] number of populated DRAM devices. 0 = N1 1 = N2

16.7.1.14 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.

Note: DEVSIZE is per subpartition.

External Memory Address Configuration, Device 0

Offset: 0x58 | Read/Write: R/W | Reset: 0x00040202 (0bxxxxxxxxxxxx0100xxxxxx10xxxxx010)

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 10 = D4096MB 11 = D8192MB 12 = D768MB
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 5 = W12

16.7.1.15 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 column width/bank width 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.

> **Note:** DEVSIZE is per subpartition.

External Memory Address Configuration, Device 1

Offset: 0x5c | Read/Write: R/W | Reset: 0x00040202 (0bxxxxxxxxxxxx0100xxxxxx10xxxxx010)

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 10 = D4096MB 11 = D8192MB 12 = D768MB
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 5 = W12

16.7.1.16 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.

The global memory client IDs with TrustZone-level security are as follows:

CLIENT	ID	ID (hex)	SWGROUP	MODULE	PARTITION	DESCRIPTION
csr_mpcorelpr	38	(0x26)	mpcorelp	mpcorelp	stop	Reads from Cortex-A15 Shadow CPU core via the L2 cache (Tegra K1 32-bit only)
csr_mpcorer	39	(0x27)	mpcore	mpcore	ftop	Reads from 4 CPU cores via the L2 cache
csw_mpcorelpw	56	(0x38)	mpcorelp	mpcorelp	stop	Writes from Cortex-A15 Shadow CPU core via the L2 cache (Tegra K1 32-bit only)
csw_mpcorew	57	(0x39)	mpcore	mpcore	ftop	Writes from 4 CPU cores via the L2 cache

Secure Region Configuration: Base

Offset: 0x70 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x80000000 (0b100000000000xxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:20	0x800	SECURITY_BOM: SECURITY_BOM is the base of the secured region, limited to MB granularity. This must point to a region of the physical address map allocated to EMEM for it to be effective; MC cannot secure address space it does not own. (In other words, this is an absolute address, not an offset.) Above is the list of clients with the TrustZone-security access. Note that AXICIF clients will adhere to the standard AXI protocol "aprot[1]==0" indication for secure requests.

16.7.1.17 MC_SECURITY_CFG1_0

Access to this register is restricted to TrustZone-secured requestors.

Boot requirements:

This security configuration register should be saved to SDRAM and restored by the OS during warm boot.

Secure Region Configuration: Bound

Offset: 0x74 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000000000000)

Bit	Reset	Description
12:0	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.

16.7.1.18 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 the top of Section 16.7.1

External Memory Arbitration Configuration

Offset: 0x90 | Read/Write: R/W | Reset: 0x00000008 (0b00000000xxx00000xxxxxxx000001000)

Bit	Reset	Description
20:16	0x0	EXTRA_TICKS_PER_UPDATE: The number of extra ticks to add every deadline timer update. Used to keep cycles-per-tick constant when the mcclk period is less than the chosen wall-clock granularity for the deadline counter. If =0, then every tick increments the deadline counter by 1; if =1, then every tick increments the deadline counter by 2, etc. Assuming a target of 30ns-per-tick, some example programming (assumes DIV=2): 10MHz emcclk => 5MHz mcclk => would need to increment deadline counter by 6.67 if using 1 cycle-per-tick => or update by 20 every 3 cycles => 3 cycles-per-tick and 19 extra ticks per update. Since performance is not critical at such slow speeds, rounding errors may be deemed acceptable, so the previous example may be simpler to program as 1 cycle-per-tick and 5 extra ticks per update. Another example, the slowest supported clock speed using a 30ns tick that doesn't distort the tick is: 1.04MHz mcclk => with 31 extra ticks per update and updating every cycle.
8:0	0x8	CYCLES_PER_UPDATE: The number of mcclk cycles per deadline timer update. The target wall-clock granularity ("tick") for the deadline counter should be fixed for the design, then the CYCLES_PER_UPDATE should be used to convert the wall-time value into mcclk cycles. All client latency allowances are also expressed in units of "ticks". Of all deadline-related controls, only the CYCLES_PER_UPDATE and EXTRA_TICKS_PER_UPDATE values need to be

Bit	Reset	Description
		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.

16.7.1.19 MC_EMEM_ARB_OUTSTANDING_REQ_0

This register can be used to limit the number of outstanding requests in the arbiter and monitor the count.

If outstanding transaction is greater than the maximum, requests are throttled based on MC_EMEM_ARB_RING1_THROTTLE_0 register

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from other MC settings.

This register is shadowed: see usage note at the top of Section 16.7.1

External Memory Arbitration Configuration: Outstanding Request Limiter

Offset: 0x94 | Read/Write: R/W | Reset: 0x8XXX0040 (0b10xxxxxxxxxxxxxxxxxxxxx001000000)

Bit	R/W	Reset	Description
31	RW	ENABLE	LIMIT_OUTSTANDING: when ENABLED, the total number of requests in the arbiter is limited to the value in ARB_MAX_OUTSTANDING 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	LIMIT_DURING_HOLDOFF_OVERRIDE: when DISABLED, override the limiting of transactions during hold-off to let requests flow freely 0 = DISABLE 1 = ENABLE
24:16	RO	X	ARB_OUTSTANDING_COUNT: Current number of outstanding requests
8:0	RW	0x40	ARB_MAX_OUTSTANDING: The maximum number of requests allowed in the arbiter when LIMIT_OUTSTANDING is set to ENABLE.

16.7.1.20 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 the top of Section 16.7.1

External Memory Arbitration Configuration: DRAM Timing: tRCD

Offset: 0x98 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

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.

16.7.1.21 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 the top of Section 16.7.1

External Memory Arbitration Configuration: DRAM Timing : tRP

Offset: 0x9c | Read/Write: R/W | Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111)

Bit	Reset	Description
6:0	0x7f	RP: The minimum number of cycles between an internal precharge command and an activate command to the same bank. Program to ceil (EMC.RP/DIV)-1+SFA. The combined value of RAP2PRE+RP 0x0 is illegal; the combined value of WAP2PRE+RP 0x0 is illegal.

16.7.1.22 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 the top of Section 16.7.1

External Memory Arbitration Configuration: DRAM Timing : tRC

Offset: 0xa0 | Read/Write: R/W | Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111)

Bit	Reset	Description
6:0	0x7f	RC: This is the minimum number of cycles between activate commands to the same bank. Program to ceil (max(EMC.RC,(EMC.RAS+EMC.RP))/DIV)-1.

16.7.1.23 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 the top of Section 16.7.1

External Memory Arbitration Configuration: DRAM Timing: tRAS

Offset: 0xa4 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

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.

16.7.1.24 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 the top of Section 16.7.1

External Memory Arbitration Configuration: DRAM Timing: tFAW

Offset: 0xa8 | Read/Write: R/W | Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111)

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.

16.7.1.25 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 the top of Section 16.7.1

External Memory Arbitration Configuration: DRAM Timing: tRRD

Offset: 0xac | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111)

Bit	Reset	Description
3:0	0xf	RRD: The minimum number of cycles between an activate command and an activate command to a different bank. Program to ceil (EMC.RRD/DIV)-1.

16.7.1.26 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 the top of Section 16.7.1

External Memory Arbitration Configuration: DRAM Timing: tRAP2PRE

Offset: 0xb0 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	RAP2PRE: The number of cycles between a read command with auto-precharge and the internal precharge command generated by the DRAM. Program to ceil (EMC_R2P/DIV). Note that: the combined value of RAP2PRE+RP 0x0 is illegal.

16.7.1.27 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 the top of Section 16.7.1

External Memory Arbitration Configuration: DRAM Timing: tWAP2PRE

Offset: 0xb4 | Read/Write: R/W | Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111)

Bit	Reset	Description
6:0	0x7f	WAP2PRE: The number of cycles between a write command with auto-precharge and the internal precharge command generated by the DRAM. Program to ceil (EMC_W2P/DIV). Note that: the combined value of WAP2PRE+RP 0x0 is illegal.

16.7.1.28 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 the top of Section 16.7.1

External Memory Arbitration Configuration: DRAM Timing: tR2R

Offset: 0xb8 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx11111)

Bit	Reset	Description
4:0	0x1f	R2R: The number of cycles between consecutive read commands to different devices (different chip selects). Program to ceil (EMC.REXT/DIV)-1+OTFA+SFA.

16.7.1.29 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 the top of Section 16.7.1

External Memory Arbitration Configuration: DRAM Timing: tW2W

Offset: 0xbc | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx11111)

Bit	Reset	Description
4:0	0x1f	W2W: The number of cycles between consecutive write commands to different devices (different chip selects). Program to ceil (EMC.WEXT/DIV)-1+SFA.

16.7.1.30 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 the top of Section 16.7.1

External Memory Arbitration Configuration: DRAM Timing: tR2W

Offset: 0xc0 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

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.

16.7.1.31 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 the top of Section 16.7.1

External Memory Arbitration Configuration: DRAM Timing: tW2R

Offset: 0xc4 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

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.

16.7.1.32 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 the top of Section 16.7.1

External Memory Arbitration Configuration: Direction Arbiter: Turns

Offset: 0xd0 | Read/Write: R/W | Reset: 0x0f0f0205 (0b00001111000011110000001000000101)

Bit	Reset	Description
31:24	0xf	W2R_TURN: Bubbles produced by a write to read bus turn. Approximately MC_EMEM_ARB_TIMING_W2R.
23:16	0xf	R2W_TURN: Bubbles produced by a read to write bus turn. Approximately MC_EMEM_ARB_TIMING_R2W.
15:8	0x2	W2W_TURN: Bubbles produced by a write to write (other device) bus turn. Approximately MC_EMEM_ARB_TIMING_W2W.
7:0	0x5	R2R_TURN: Bubbles produced by a read to read (other device) bus turn. Approximately MC_EMEM_ARB_TIMING_R2R.

16.7.1.33 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 the top of Section 16.7.1

External Memory Arbitration Configuration: Direction Arbiter: Covers

Offset: 0xd4 | Read/Write: R/W | Reset: 0x000f0f0f (0bxxxxxxxx00001111000011110000111 1)

Bit	Reset	Description
23:16	0xf	RCD_W_COVER: Cost to cover the activate-to-write delay. Approximately ceil ((EMC.RTP + EMC.RP + EMC.RD_RCD)/DIV)-1.
15:8	0xf	RCD_R_COVER: Cost to cover the activate-to-read delay. Approximately ceil ((EMC.WTP + EMC.RP + EMC.WR_RCD)/DIV)-1.
7:0	0xf	RC_COVER: Cost to cover the activate-to-activate delay. Approximately MC_EMEM_ARB_TIMING_RC.

16.7.1.34 MC_EMEM_ARB_MISC0_0

BC2AA_HOLDOFF: In the case where the currently-open pages have more work pending than the time it would take to open another page, it is better to hold off opening that page until it is truly needed. The Best-bank Cache computes the pending work and compares it to this threshold and advises the Activation Arbiter to hold-off.

Priority Inversion: In general, if the arbiter is working on a lower-priority request that blocks a higher-priority request, these thresholds are used to limit the number of low-priority requests we will service before the arbiter will interrupt the lower-priority traffic to start servicing the higher-priority traffic.

The priority classes are (in increasing priority order): Low (! expired), High (expired), HighIso.

There are three ways priority can be inverted: Bank Activation, Bus Direction, and Refresh:

Bank Activation: If an unexpired request to bank A, row R currently holds the page open, and a request to bank A, row S expires, the Transaction Arbiter must make a decision whether to finish out the work for row R or to interrupt that transfer to immediately service row S. The TA makes this decision based on the remaining work for row R and whether row S is for an isochronous client, and the two PRIORITY_INVERSION thresholds.

Bus Direction: If the Transaction Arbiter is currently working on expired requests in direction P, and requests are also expired (or expired-iso) in direction Q, the Transaction Arbiter must make a decision whether to continue working in direction P or switch the bus to direction Q. The Transaction Arbiter will service at maximum PRIORITY_INVERSION_THRESHOLD (or _INVERSION_ISO_THRESHOLD) requests in direction P before switching to direction Q.

Refresh: If the Transaction Arbiter is currently working on requests, and EMC asks to inject a refresh, the Transaction Arbiter must decide whether it is more efficient (overall) to close the pages immediately or to finish the pages and then close them. Depending on the setting of the REFRESH_ACK_THRESHOLD_USAGE configuration bit (in EMEM_ARB_MISC1 register, below), the Transaction Arbiter will either choose to close the page immediately (NONE), close the page immediately if its length is over the iso threshold (ISO), or close the page immediately if its length is over the normal threshold (NORMAL). If the length is less than the chosen threshold, the Transaction Arbiter will service all remaining requests to the page before acknowledging the EMC refresh request.

When in DDR3 coalesce-for-performance mode, the arbiters will monitor the traffic patterns and might 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 the top of Section 16.7.1

External Memory Arbitration Configuration: Miscellaneous Thresholds (0)

Offset: 0xd8 | Read/Write: R/W | Reset: 0x720a4010 (0bx11100100000101001000000000010000)

Bit	Reset	Description
30:28	0x7	ATOMS_PER_DVFS_PULSE: [PMC] ATOMS_PER_DVFS_PULSE is used to control the rate at which the MC updates the sys_stats_mon outputs. The MC will toggle the sys_stats_mon outputs for every 2^(ATOMS_PER_DVFS_PULSE+1) atoms of traffic, where an atom is defined as any transaction. Note that 16B, 32B, and 64B requests consume the same amount of memory bus resources. So, in this context, an atom is defined as any request and not cumulative 64B sized transactions. The maximum ATOMS_PER_DVFS_PULSEsetting supported is the reset value, also known as NV_MC_EMEM_DVFS_CNTR_CFG_RESET.
27	DISABLE	MC_EMC_SAME_FREQ: [PMC] Used in conjunction with EMEM_ARB_MISC1.COALESCE_FOR_PERFORMANCE to configure how often the 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 the MC can send a data request every cycle. If configured to DIV=1 BL=4, then the 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 the MC can send a data request once every four cycles. 0 = MC is 1/2 the frequency of EMC (DIV=2). 1 = MC is the same frequency of EMC (DIV=1). 0 = DISABLE 1 = ENABLE
26:21	0x10	EXPIRING_SOON_SLACK_THRESHOLD: Slack threshold below which an isochronous request is considered to be expiring "soon". Used to de-assert mc2emc_idle signal early to wake EMC out of power-down or self-refresh. If using EMC's Dynamic Self Refresh features, set this to the ticks needed to cover exit self-refresh delay; otherwise, if using EMC's ACPD feature, set to the ticks needed to cover exit power-down delay; if neither power-saving feature is being used, the programming of this threshold should have no effect.

Bit	Reset	Description
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.
15	DISABLE	EMC_REQ_B2B_XFER: When enabled, the MC allows EMC transactions on back-to-back clocks regardless of EMC consumption bandwidth 0 = DISABLE 1 = ENABLE
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.

16.7.1.35 MC_EMEM_ARB_MISC1_0

Two DVFS counters exist in the "NV_MC_ARB_EMEM_stats.v" module: one for all requests and another for CPU-only transactions. Both of these are down counters which decrement for each transaction atom on mcclk.

When these counters reach zero, a single mcclk pulse is generated, and the counters reinitialize to a value of ((2^(ATOMS_PER_DVFS_PULSE+1))-1). The single mcclk pulse that is generated (when the counters reach zero) gets synchronized to sclk (system clock) through a cross-clock domain FIFO control module, which has a minimum response time requirement of (4 * sclk_period + 4 * mcclk_period). Therefore, ATOMS_PER_DVFS_PULSE should never be set such that ((2^(ATOMS_PER_DVFS_PULSE+1)) * mcclk_period) is less than (4 * sclk_period + 4 * mcclk_period).

This translates to the following rule for setting the minimum value of ATOMS_PER_DVFS_PULSE:

ATOMS_PER_DVFS_PULSE >= log2((4 * sclk_period/mcclk_period) + 4) - 1

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.

This register is shadowed: see usage note at the top of Section 16.7.1

External Memory Arbitration Configuration: Miscellaneous Thresholds

Offset: 0xdc | Read/Write: R/W | Reset: 0x80a00002 (0b1000xx00101xxxxxxxx000000000xx10) | Default: 0x00000000

Bit	Reset	SW Default	Description
31:28	0x8	NONE	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 greater than or equal to NV_MC_EMEM_DL_WIDTH is UNDEFINED. If the threshold is reached, ARBITRATION_EMEM_INT will fire.
25:24	NORMAL	NONE	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
23:21	USE_EXPIRING_SOON_SLACK_THRESHOLD	NONE	EXPIRING_SOON_SLACK_THRESHOLD_PD: [PMC] Slack threshold below which an isochronous request is considered to be expiring "soon". Used to deassert the mc2emc_idle_pd signal early to wake the EMC out of power-down. (EXPIRING_SOON_SLACK_THRESHOLD controls exit from self-refresh.) This register should be set to the ticks needed to cover the exit power-down delay. If the available choices are not sufficient, the value should be set in EXPIRING_SOON_SLACK_THRESHOLD and this register should be set to USE_EXPIRING_SOON_SLACK_THRESHOLD 0 = ZERO_TICKS : 0 tick threshold for PD expiring soon 1 = ONE_TICK: 1 tick threshold for PD expiring soon 2 = TWO_TICKS: 2 tick threshold for PD expiring soon 3 = FOUR_TICKS: 4 tick threshold for PD expiring soon 4 = EIGHT_TICKS: 8 tick threshold for PD expiring soon 5 = USE_EXPIRING_SOON_SLACK_THRESHOLD: Use the setting in EXPIRING_SOON_SLACK_THRESHOLD to control mc2emc_idle_pd
12:4	0x0	0x0	ALT_DEADLOCK_PREVENTION_SLACK_THRESHOLD: [PMC] Use the absolute value of deadlock prevention slack threshold instead of 2^n format. Note that the DEADLOCK_PREVENTION_SLACK_THRESHOLD field still needs to be programmed along with this one (for example, 7).
1	ENABLE	NONE	ALLOW_BCA_HOLDOFF_WHEN_EXP: [PMC] When enabled, allows BCA holdoff to occur even when expired requests are present. 0 = DISABLE 1 = ENABLE
0	DISABLE	0x0	BLOCK_LP_CPU_RD_IF_SMMU_INP_HP: [PMC] When enabled, low priority (due to DDA) CPU reads get blocked when the SMMU output is high priority and a request is present at the input to the SMMU. 0 = DISABLE 1 = ENABLE

16.7.1.36 MC_EMEM_ARB_RING1_THROTTLE_0

External Memory Arbitration Configuration: Snap Arbiter Throttle

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 the top of Section 16.7.1

Offset: 0xe0 | Read/Write: R/W | Reset: 0x001f0000 (0bxxxxxxxxxxxx11111xxxxxxxxxxx00000)

Bit	Reset	Description
20:16	0x1f	RING1_THROTTLE_CYCLES_HIGH: Cycles of throttle after each request when outstanding transaction count is greater than or equal to threshold.
4:0	0x0	RING1_THROTTLE_CYCLES_LOW: Cycles of throttle after each request when outstanding transaction count is below threshold. Suggested programming: (DIV==1)? 1 : 0.

16.7.1.37 MC_EMEM_ARB_RING3_THROTTLE_0

External Memory Arbitration Configuration: Snap Arbiter Throttle

Highest Ring Snap Arbiter Throttle - Although called ring3, this register works on the highest ring.

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 the top of Section 16.7.1

Offset: 0xe4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000xxxxxxxxxxx00000)

Bit	Reset	Description
4:0	0x0	RING3_THROTTLE_CYCLES: Cycles of throttle after each request.

16.7.1.38 MC_EMEM_ARB_OVERRIDE_0

External Memory Arbitration Configuration: Feature Overrides

Boot requirements:

- Some fields of this register should be saved in the scratch registers and restored by the Boot ROM during warm boot.
- This register should be saved in the SDRAM and restored by the OS during warm boot.

Offset: 0xe8 | Read/Write: R/W | Reset: 0x10000000 (0b0xx100xx00000000000000000xxx0xxxx)

Bit	Reset	Description
31	DISABLE	ARB_HUM_FIFO_DEADLOCK_CHECK_OVERRIDE 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
28	ENABLE	ARB_EMEM_SPO_OVERRIDE 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
27	DISABLE	ARB_EMEM_AP_OVERRIDE 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
		1 = OVERRIDE
26	DISABLE	ARB_HUM_FIFO_OVERRIDE 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
23	DISABLE	ISO_TA_OVERRIDE 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
22	DISABLE	ISO_DA_OVERRIDE 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
21	DISABLE	ISO_BC_INHERIT_ON_PRIINV_OVERRIDE 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
20	DISABLE	ISO_BC_CAUSE_PRIINV_OVERRIDE 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
19	DISABLE	ISO_BA_OVERRIDE 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
18	DISABLE	ISO_AA_OVERRIDE 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
17	DISABLE	ARB_EMEM_BUBBLECALC_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
16	DISABLE	ALLOC_ONE_BQ_PER_CLIENT: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
15	DISABLE	PRIORITY_INVERSION_ISO_THRESHOLD_BUS_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
14	DISABLE	PRIORITY_INVERSION_ISO_THRESHOLD_BANK_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
13	DISABLE	PRIORITY_INVERSION_THRESHOLD_BUS_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
12	DISABLE	PRIORITY_INVERSION_THRESHOLD_BANK_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
11	DISABLE	PRIORITY_INVERSION_EQ_PRI_LEN_LIMIT_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE

Bit	Reset	Description
10	DISABLE	OBSERVED_DIRECTION_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
9	DISABLE	BC2AA_HOLDOFF_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
8	DISABLE	TS2AA_HOLDOFF_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
4	DISABLE	EXPIRE_UPDATE_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE

16.7.1.39 MC_EMEM_ARB_RSV_0

EMEM Arbiter Reserved

Boot requirements:

- Some fields of this register should be saved in the scratch registers and restored by the Boot ROM during warm boot.
- This register should be saved in the SDRAM and restored by the OS during warm boot.

This register is shadowed: see usage note at the top of Section 16.7.1

Offset: 0xec | Read/Write: R/W | Reset: 0xff00ff00 (0b11111111000000001111111100000000)

Bit	Reset	Description
31:24	0xff	EMEM_ARB_RESERVED_BYTE3
23:16	0x0	EMEM_ARB_RESERVED_BYTE2
15:8	0xff	EMEM_ARB_RESERVED_BYTE1
7:0	0x0	EMEM_ARB_RESERVED_BYTE0

16.7.1.40 MC_CLKEN_OVERRIDE_0

Second-Level Clock Enable Overrides

Boot requirements: This register should be saved in the SDRAM and restored by the OS during warm boot.

Offset: 0xf4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000x00x0)

Bit	Reset	Description
9	CLK_GATED	WCAM_CLKEN_OVR : 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
8	CLK_GATED	PTC_CACHE_CLKEN_OVR : 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE 0 = DISABLED 1 = ENABLED
7	CLK_GATED	PTC_CLKEN_OVR : 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
6	CLK_GATED	TLB_CLKEN_OVR : 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
5	CLK_GATED	WB_CLKEN_OVR : 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
3	CLK_GATED	REGS_CLKEN_OVR : 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
2	CLK_GATED	EARB_CLKEN_OVR : 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
0	CLK_GATED	CIF_CLKEN_OVR : 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED

16.7.1.41 MC_TIMING_CONTROL_0

Shadowed Registers: Update Trigger

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: 0xfc | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	TIMING_UPDATE

16.7.1.42 MC_STAT_CONTROL_0

Statistics: Control

Offset: 0x100 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	DISABLE	EMC_PM_STOP_TRIGGER: 0 = DISABLE 1 = ENABLE
2	DISABLE	EMC_PM_START_TRIGGER: 0 = DISABLE 1 = ENABLE
1:0	RST	EMC_GATHER: 0 = RST 2 = DISABLE 3 = ENABLE

16.7.1.43 MC_CLIENT_HOTRESET_CTRL_0

Writing FLUSH_ENABLE to a bit in this register causes a flush to be performed on all of the clients for the selected swname. The status of the flush (done/in progress) can be monitored in the CLIENT_HOTRESET_STATUS register.

A proper client reset sequence is as follows:

1. Set the appropriate FLUSH_ENABLE bit in the CLIENT_HOTRESET_CTRL register to block further access by the client, and start the flush process.
2. Poll the CLIENT_HOTRESET_STATUS register until the appropriate bit returns FLUSH_DONE.
3. Clear module bit in the CLK_RST_CONTROLLER_RST_DEVICES_* register to reset the module.
4. Set module bit in the CLK_RST_CONTROLLER_RST_DEVICES_* register to release the module reset.
5. Clear the appropriate FLUSH_ENABLE bit in the CLIENT_HOTRESET_CTRL register to allow transactions to flow.

Memory Client Hot Reset Control Register

Offset: 0x200 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxx000000000xx000000xx0000

Bit	Reset	Description
31	DISABLE	SDMMC3A_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
30	DISABLE	SDMMC2A_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
29	DISABLE	SDMMC1A_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
22	DISABLE	TSEC_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
20	DISABLE	XUSB_DEV_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
19	DISABLE	XUSB_HOST_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
18	DISABLE	VIC_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
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
11	DISABLE	MSENC _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	ISP2_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
7	DISABLE	HDA_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED

Bit	Reset	Description
		1 = ENABLED
6	DISABLE	HC_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

16.7.1.44 MC_CLIENT_HOTRESET_STATUS_0

Contains one bit for each swname, indicating the status of any flush that has been requested in the CLIENT_HOTRESET_CTRL register.

Note: If no flush has been requested, this register returns FLUSH_DONE.

Memory Client Hot Reset Status Register

Offset: 0x204 | Read/Write: RO | Reset: 0xX0XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SDMMC3A_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
30	X	SDMMC2A_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
29	X	SDMMC1A_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
22	X	TSEC_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
20	X	XUSB_DEV_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
19	X	XUSB_HOST_HOTRESET_STATUS: 1 = FLUSH_DONE

Bit	Reset	Description
		0 = FLUSH_IN_PROGRESS
18	X	VIC_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
17	X	VI_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
16	X	VDE_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
15	X	SATA_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
14	X	PPCS_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
11	X	MSENC_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	ISP2_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
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

16.7.1.45 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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-Client Isochronous Settings

Offset: 0x208 | Read/Write: R/W | Reset: 0x0023007e (0b0000xxxx001xxx1100xxxxxxx1111110

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_MSENCSRD: client msencsrd 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
17	0x1	ISO_DISPLAYHCB: client displayhcb is treated as an isochronous client 0 = DISABLE 1 = ENABLE
16	0x1	ISO_DISPLAYHC: client displayhc is treated as an isochronous client 0 = DISABLE 1 = ENABLE
15	DISABLE	ISO_AVPCARM7R: client avpcarm7r is treated as an isochronous client 0 = DISABLE 1 = ENABLE
14	DISABLE	ISO_AFIR: client afir is treated as an isochronous client 0 = DISABLE 1 = ENABLE
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

Bit	Reset	Description
		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

16.7.1.46 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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-client isochronous settings

Offset: 0x20c | Read/Write: R/W | Reset: 0x00000000 (0b00000x00x00xx00xxxxx0xxx000000xx

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
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
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
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

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
11	DISABLE	ISO_MSENCSWR: client msencswr 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

16.7.1.47 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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-Client Isochronous Settings

Offset: 0x210 | Read/Write: R/W | Reset: 0x04000000 (0bxxxxx1000000xx00x00000xx00x0xx00

Bit	Reset	Description
26	0x1	ISO_DISPLAYT: Client displayt is treated as an isochronous client 0 = DISABLE 1 = ENABLE
25	DISABLE	ISO_GPUSWR: Client gpuswr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
24	DISABLE	ISO_GPUSRD: Client gpusrd is treated as an isochronous client 0 = DISABLE 1 = ENABLE
21	DISABLE	ISO_TSECSWR: Client tsecswr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
20	DISABLE	ISO_TSECSRD: Client tsecsrd is treated as an isochronous client 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
17	DISABLE	ISO_ISPWBB: Client emucifwispwbb is treated as an isochronous client 0 = DISABLE 1 = ENABLE
16	DISABLE	ISO_ISPWAB: Client ispwab is treated as an isochronous client 0 = DISABLE 1 = ENABLE
14	DISABLE	ISO_ISPRAB: Client isprab is treated as an isochronous client 0 = DISABLE 1 = ENABLE
13	DISABLE	ISO_XUSB_DEVW: Client xusb_devw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
12	DISABLE	ISO_XUSB_DEVR: Client xusb_devr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
11	DISABLE	ISO_XUSB_HOSTW: Client xusb_hostw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
10	DISABLE	ISO_XUSB_HOSTR: Client xusb_hostr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
7	DISABLE	ISO_ISPWB: Client ispwb is treated as an isochronous client 0 = DISABLE 1 = ENABLE
6	DISABLE	ISO_ISPWA: Client ispwa is treated as an isochronous client 0 = DISABLE 1 = ENABLE
4	DISABLE	ISO_ISPRA: Client ispra is treated as an isochronous client 0 = DISABLE 1 = ENABLE
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

16.7.1.48 MC_EMEM_ARB_ISOCHRONOUS_3_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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-Client Isochronous Settings

Offset: 0x214 | Read/Write: R/W | Reset: 0x00080000 (0bxxxxxxxxxxxx10xxxx00xxxx00000000)

Bit	Reset	Description
19	0x1	ISO_DISPLAYD: Client displayd is treated as an isochronous client 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
18	DISABLE	ISO_VIW: Client viw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
13	DISABLE	ISO_VICSWR: Client vicswr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
12	DISABLE	ISO_VICSRD: Client vicsrd is treated as an isochronous client 0 = DISABLE 1 = ENABLE
7	DISABLE	ISO_SDMMCWAB: Client sdmmcwab is treated as an isochronous client 0 = DISABLE 1 = ENABLE
6	DISABLE	ISO_SDMMCW: Client sdmmcw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
5	DISABLE	ISO_SDMMCWAA: Client sdmmcwaa is treated as an isochronous client 0 = DISABLE 1 = ENABLE
4	DISABLE	ISO_SDMMCWA: Client sdmmcwa is treated as an isochronous client 0 = DISABLE 1 = ENABLE
3	DISABLE	ISO_SDMMCRAB: Client sdmmcrab is treated as an isochronous client 0 = DISABLE 1 = ENABLE
2	DISABLE	ISO_SDMMCR: Client sdmmcr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
1	DISABLE	ISO_SDMMCRAA: Client sdmmcraa is treated as an isochronous client 0 = DISABLE 1 = ENABLE
0	DISABLE	ISO_SDMMCRA: Client sdmmcra is treated as an isochronous client 0 = DISABLE 1 = ENABLE

16.7.1.49 MC_EMEM_ARB_HYSTERESIS_0_0

Controls whether a client is considered to be a hysteresis client for EMEM arbitration. If a client is NOT marked hysteresis, it will immediately cause the arbiter to turn off ts2aa_holdoff (see the comments for EMEM_ARB_OVERRIDE for further description of ts2aa_holdoff functionality).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-Client Hysteresis Settings

Offset: 0x218 | Read/Write: R/W | Reset: 0x0003007e (0b0000xxxx000xxx1100xxxxxxx1111110)

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

Bit	Reset	Description
		1 = ENABLE
29	DISABLE	HYST_PPCSAHBDMAR: client ppcsahbdmar is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
28	DISABLE	HYST_MSENCSRD: client msencsrd is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
23	DISABLE	HYST_HOST1XR: client host1xr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
22	DISABLE	HYST_HOST1XDMAR: client host1xdmar is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
21	DISABLE	HYST_HDAR: client hdar is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
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
6	0x1	HYST_DISPLAY0CB: client display0cb is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
5	0x1	HYST_DISPLAY0C: client display0c is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
4	0x1	HYST_DISPLAY0BB: client display0bb is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
3	0x1	HYST_DISPLAY0B: client display0b is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
2	0x1	HYST_DISPLAY0AB: client display0ab is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
1	0x1	HYST_DISPLAY0A: client display0a is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
0	DISABLE	HYST_PTCR: client ptcr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE

16.7.1.50 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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-Client Hysteresis Settings

Offset: 0x21c | Read/Write: R/W | Reset: 0x00000000 (0b00000x00x00xx00xxxxx0xxx000000xx

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
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
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
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
11	DISABLE	HYST_ MSENCSWR: client msencswr 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

Bit	Reset	Description
5	DISABLE	HYST_VDETPER: client vdetper is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
4	DISABLE	HYST_VDEMCER: client vdemcer is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
3	DISABLE	HYST_VDEMBER: client vdember is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
2	DISABLE	HYST_VDEBSEVR: client vdebsevr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE

16.7.1.51 MC_EMEM_ARB_HYSTERESIS_2_0

Controls whether a client is considered to be a hysteresis client for EMEM arbitration. If a client is NOT marked hysteresis, it will immediately cause the arbiter to turn off ts2aa_holdoff (see the comments for EMEM_ARB_OVERRIDE for further description of ts2aa_holdoff functionality).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-client hysteresis settings

Offset: 0x220 | Read/Write: R/W | Reset: 0x04000000 (0bxxxxx1000000xx00x00000xx00x0xx00)

Bit	Reset	Description
26	0x1	HYST_DISPLAYT: client displayt is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
25	DISABLE	HYST_GPUSWR: client gpuswr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
24	DISABLE	HYST_GPUSRD: client gpusrd is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
21	DISABLE	HYST_TSECSWR: client tsecswr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
20	DISABLE	HYST_TSECSRD: client tsecsrd is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
17	DISABLE	HYST_ISPWBB: client ispwbb is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
16	DISABLE	HYST_ISPWAB: client ispwab is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
14	DISABLE	HYST_ISPRAB: client isprab is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
13	DISABLE	HYST_XUSB_DEVW: client xusb_devw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
12	DISABLE	HYST_XUSB_DEVR: client xusb_devr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
11	DISABLE	HYST_XUSB_HOSTW: client xusb_hostw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
10	DISABLE	HYST_XUSB_HOSTR: client xusb_hostr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
7	DISABLE	HYST_ISPWB: client ispwb is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
6	DISABLE	HYST_ISPWA: client ispwa is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
4	DISABLE	HYST_ISPRA: client ispra is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
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

16.7.1.52 MC_EMEM_ARB_HYSTERESIS_3_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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-client hysteresis settings

Offset: 0x224 | Read/Write: R/W | Reset: 0x00080000 (0bxxxxxxxxxxxx10xxxx00xxxx00000000)

Bit	Reset	Description
19	0x1	HYST_DISPLAYD: client displayd is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
18	DISABLE	HYST_VIW: client viw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
13	DISABLE	HYST_VICSWR: client vicswr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
12	DISABLE	HYST_VICSRD: client vicsrd is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
7	DISABLE	HYST_SDMMCWAB: client sdmmcwab is treated as a hysteresis client 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	DISABLE	HYST_SDMMCW: client sdmmcw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
5	DISABLE	HYST_SDMMCWAA: client sdmmcwaa is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
4	DISABLE	HYST_SDMMCWA: client sdmmcwa is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
3	DISABLE	HYST_SDMMCRAB: client sdmmcrab is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
2	DISABLE	HYST_SDMMCR: client sdmmcr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
1	DISABLE	HYST_SDMMCRAA: client sdmmcraa is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
0	DISABLE	HYST_SDMMCRA: client sdmmcra is treated as a hysteresis client 0 = DISABLE 1 = ENABLE

16.7.1.53 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.

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-client SMMU translation enables

Offset: 0x228 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xf0e3c07e (0b1111xxxx111xxx1111xxxxxxx111111x)

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
28	ENABLE	SMMU_MSENCSRD_ENABLE: enable client msencsrd 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

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
21	ENABLE	SMMU_HDAR_ENABLE: enable client hdar 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
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

16.7.1.54 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.

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-client SMMU translation enables

Offset: 0x22c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xf866083c (0b11111xxxx11xx11xxxxx1xxxxx1111xx

Bit	Reset	Description
31	ENABLE	SMMU_VDEDBGW_ENABLE: enable client vdedbgw to be translated by SMMU 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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
27	ENABLE	SMMU_PPCSAHBDMAW_ENABLE: enable client ppcsahbdmaw 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
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
11	ENABLE	SMMU_MSENCSWR _ENABLE: enable client msencswr to be translated by SMMU 0 = DISABLE 1 = ENABLE
5	ENABLE	SMMU_VDETPER_ENABLE: enable client vdetper to be translated by SMMU 0 = DISABLE 1 = ENABLE
4	ENABLE	SMMU_VDEMCER_ENABLE: enable client vdemcer to be translated by SMMU 0 = DISABLE 1 = ENABLE
3	ENABLE	SMMU_VDEMBER_ENABLE: enable client vdember to be translated by SMMU 0 = DISABLE 1 = ENABLE
2	ENABLE	SMMU_VDEBSEVR_ENABLE: enable client vdebsevr to be translated by SMMU 0 = DISABLE 1 = ENABLE

16.7.1.55 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.

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-client SMMU translation enables

Offset: 0x230 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0Xf37cd3 (0bxxxxx1xx1111xx11x11111xx11x1xx11)

Bit	R/W	Reset	Description
26	RW	ENABLE	SMMU_DISPLAYT_ENABLE: enable client displayt to be translated by the SMMU 0 = DISABLE 1 = ENABLE
25	RO	X	SMMU_GPUSWR_ENABLE: GPU translation is handled by swid, so this is required to be enabled 0 = DISABLE 1 = ENABLE
24	RO	X	SMMU_GPUSRD_ENABLE: GPU translation is handled by swid, so this is required to be enabled 0 = DISABLE 1 = ENABLE
21	RW	ENABLE	SMMU_TSECSWR_ENABLE: enable client tsecswr to be translated by the SMMU 0 = DISABLE 1 = ENABLE
20	RW	ENABLE	MMU_TSECSRD_ENABLE: enable client tsecsrd to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
17	RW	ENABLE	SMMU_ISPWBB_ENABLE: enable client ispwbb to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
16	RW	ENABLE	SMMU_ISPWAB_ENABLE: enable client ispwab to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
14	RW	ENABLE	SMMU_ISPRAB_ENABLE: enable client isprab to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
13	RW	ENABLE	SMMU_XUSB_DEVW_ENABLE: enable client xusb_devw to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
12	RW	ENABLE	SMMU_XUSB_DEVR_ENABLE: enable client xusb_devr to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
11	RW	ENABLE	SMMU_XUSB_HOSTW_ENABLE: enable client xusb_hostw to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
10	RW	ENABLE	SMMU_XUSB_HOSTR_ENABLE: enable client xusb_hostr to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
7	RW	ENABLE	SMMU_ISPWB_ENABLE: enable client ispwb to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
6	RW	ENABLE	SMMU_ISPWA_ENABLE: enable client ispwa to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
4	RW	ENABLE	SMMU_ISPRA_ENABLE: enable client ispra to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
1	RW	ENABLE	SMMU_VDETPMW_ENABLE: enable client vdetpmw to be translated by SMMU 0 = DISABLE 1 = ENABLE
0	RW	ENABLE	SMMU_VDEMBEW_ENABLE: enable client vdembew to be translated by SMMU 0 = DISABLE 1 = ENABLE

16.7.1.56 MC_SMMU_TRANSLATION_ENABLE_3_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.

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-Client SMMU Translation Enables

Offset: 0x234 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x000c30ff (0bxxxxxxxxxxxxx11xxxx11xxxx11111111)

Bit	Reset	Description
19	ENABLE	SMMU_DISPLAYD_ENABLE: enable client displayd to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
18	ENABLE	SMMU_VIW_ENABLE: enable client viw to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
13	ENABLE	SMMU_VICSRD_ENABLE: enable client vicsrd to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
12	ENABLE	SMMU_VICSRD_ENABLE: enable client vicsrd to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
7	ENABLE	SMMU_SDMMCWAB_ENABLE: enable client sdmmcwab to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
6	ENABLE	SMMU_SDMMCW_ENABLE: enable client sdmmcw to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
5	ENABLE	SMMU_SDMMCWAA_ENABLE: enable client sdmmcwaa to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
4	ENABLE	SMMU_SDMMCWA_ENABLE: enable client sdmmcwa to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
3	ENABLE	SMMU_SDMMCRAB_ENABLE: enable client sdmmcrab to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
2	ENABLE	SMMU_SDMMCR_ENABLE: enable client sdmmcr to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
1	ENABLE	SMMU_SDMMCRAA_ENABLE: enable client sdmmcraa to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
0	0x0	SMMU_SDMMCRA_ENABLE: enable client sdmmcra to be translated by the SMMU. 0 = DISABLE 1 = ENABLE

16.7.1.57 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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU AFI ASID Assignment Register

Offset: 0x238 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	AFI_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	AFI_ASID: ASID to use for translation, if enabled

16.7.1.58 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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU AVPC ASID Assignment Register

Offset: 0x23c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	AVPC_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	AVPC_ASID: ASID to use for translation, if enabled

16.7.1.59 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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU DC ASID Assignment Register

Offset: 0x240 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	DC_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	DC_ASID: ASID to use for translation, if enabled

16.7.1.60 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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU DCB ASID Assignment Register

Offset: 0x244 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	DCB_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	DCB_ASID: ASID to use for translation, if enabled

16.7.1.61 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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU HC ASID Assignment Register

Offset: 0x250 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	HC_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	HC_ASID: ASID to use for translation, if enabled

16.7.1.62 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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU HDA ASID Assignment Register

Offset: 0x254 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	HDA_SMMU_ENABLE: if set, translation is enabled for this module 0 = DISABLE 1 = ENABLE
6:0	0x0	HDA_ASID: ASID to use for translation, if enabled

16.7.1.63 MC_SMMU_ISP2_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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU ISP2 ASID Assignment Register

Offset: 0x258 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	ISP2_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	ISP2_ASID: ASID to use for translation, if enabled

16.7.1.64 MC_SMMU_MSENC_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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU MSENC ASID Assignment Register

Offset: 0x264 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	MSENC_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	MSENC_ASID: ASID to use for translation, if enabled

16.7.1.65 MC_SMMU_NV_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field.

SMMU NV ASID Assignment Register

Offset: 0x268 | Read/Write: R/W | Reset: 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

16.7.1.66 MC_SMMU_NV2_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field.

SMMU NV2 ASID Assignment Register

Offset: 0x26c | Read/Write: R/W | Reset: 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

16.7.1.67 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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU PPCS ASID Assignment Register

Offset: 0x270 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	PPCS_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	PPCS_ASID: ASID to use for translation, if enabled

16.7.1.68 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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU SATA ASID Assignment Register

Offset: 0x274 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	SATA_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	SATA_ASID: ASID to use for translation, if enabled

16.7.1.69 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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU VDE ASID Assignment Register

Offset: 0x27c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	VDE_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
1:0	0x0	VDE_ASID: ASID to use for translation, if enabled

16.7.1.70 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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU VI ASID Assignment Register

Offset: 0x280 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	VI_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	VI_ASID: ASID to use for translation, if enabled

16.7.1.71 MC_SMMU_VIC_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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU VIC ASID Assignment Register

Offset: 0x284 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	VIC_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	VIC_ASID: ASID to use for translation, if enabled

16.7.1.72 MC_SMMU_XUSB_HOST_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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU XUSB_HOST ASID Assignment Register

Offset: 0x288 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	XUSB_HOST_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	XUSB_HOST_ASID: ASID to use for translation, if enabled

16.7.1.73 MC_SMMU_XUSB_DEV_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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU XUSB_DEV ASID Assignment Register

Offset: 0x28c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	XUSB_DEV_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	XUSB_DEV_ASID: ASID to use for translation, if enabled

16.7.1.74 MC_SMMU_TSEC_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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU TSEC ASID Assignment Register

Offset: 0x294 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	TSEC_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	TSEC_ASID: ASID to use for translation, if enabled

16.7.1.75 MC_SMMU_PPCS1_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).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU PPCS1ASID Assignment Register

Offset: 0x298 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	PPCS1_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	PPCS1_ASID: ASID to use for translation, if enabled

16.7.1.76 MC_VIDEO_PROTECT_VPR_OVERRIDE_0

MC VPR OVERRIDE Register

Offset: 0x418 | Read/Write: R/W | Reset: 0xe4bac743 (0b111xx1xx1011101011xx011101xx0011)

Bit	Reset	Description
31	ENABLE	SDMMC3A_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
30	ENABLE	SDMMC2A_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
29	ENABLE	SDMMC1A_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
26	ENABLE	DC1_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
23	ENABLE	PPCS1_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
22	DISABLE	TSEC_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
20	ENABLE	XUSB_DEV_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
19	ENABLE	XUSB_HOST_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
18	DISABLE	VIC_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
17	ENABLE	VI_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
16	DISABLE	VDE_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
15	ENABLE	SATA_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
14	ENABLE	PPCS_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
11	DISABLE	MSENC_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
10	ENABLE	MPCORELP_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
9	ENABLE	MPCORE_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
8	ENABLE	ISP2_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
7	DISABLE	HDA_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
6	ENABLE	HC_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
3	DISABLE	DCB_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
2	DISABLE	DC_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	ENABLE	AVPC_VPR_OVERRIDE 0 = DISABLE 1 = ENABLE
0	ENABLE	AFI_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE

16.7.1.77 MC_VIDEO_PROTECT_VPR_OVERRIDE1_0

MC VPR OVERRIDE Register

Offset: 0x590 | Read/Write: R/W | Reset: 0x00000013 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx10011)

Bit	Reset	Description
4	ENABLE	PPCS2_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
3	DISABLE	GPUB_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
2	DISABLE	GPU_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
1	ENABLE	ISP2B_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
0	ENABLE	SDMMC4A_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE

16.7.1.78 MC_SMMU_TLB_SET_SELECTION_MASK_0_0

TLB Set Selection Mask (Bit 0)

Mask is ANDed with the Virtual Address, and the resulting value is XORed to a single bit. That bit selects either set0 or set1.

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 the top of Section 16.7.1

Offset: 0x600 | Read/Write: R/W | Reset: 0x00008000 (0bxxxxxxxx000000001xxxxxxxxxxxxxxx)

Bit	Reset	Description
23:15	0x1	TLB_SET_SELECTION_MASK_0

16.7.1.79 MC_DISPLAY_SNAP_RING_0

Display Arbiter Ring Selection

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Offset: 0x608 | Read/Write: R/W | Reset: 0x00000003 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx11)

Bit	Reset	Description
31	ENABLED	DISPLAY_SNAP_RING_WRITE_ACCESS : 0 = ENABLED 1 = DISABLED
1	RING1	DISB_SNAP_RING : 0 = RING0 1 = RING1
0	RING1	DIS_SNAP_RING : 0 = RING0 1 = RING1

16.7.1.80 MC_ERR_VPR_STATUS_0

Offset: 0x654 | Read/Write: RO | Reset: 0x00XXX0XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
21:20	X	ERR_VPR_ADR_HI: Higher address bits of erring address whose lower bits are available in the ERR_VPR_ADR register
18	X	ERR_VPR_SWAP
17	X	ERR_VPR_SECURITY: Set if transaction was secure. 0 = NONSECURE 1 = SECURE
16	X	ERR_VPR_RW: Direction of the access that caused the error. 0 = READ 1 = WRITE
14:12	X	ERR_VPR_ADR1: Second subpartition unique address bits
6:0	X	ERR_VPR_ID: Client identifier (see above list) of the access that caused the error.

16.7.1.81 MC_ERR_VPR_ADR_0

Offset: 0x658 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ERR_VPR_ADR

16.7.1.82 MC_EMEM_CFG_ACCESS_CTRL_0

Access Control Bit for EMEM_CFG Registers

This bit is "sticky." Registers are writeable by default. Once write access is disabled, it cannot be re-enabled without a system reset.

The following registers are controlled by this register:

- EMEM_CFG
- EMEM_ADR_CFG

- EMEM_ADR_CFG_DEV0
- EMEM_ADR_CFG_DEV1
- EMEM_ADR_CFG_CHANNEL_MASK
- EMEM_ADR_CFG_BANK_MASK_0
- EMEM_ADR_CFG_BANK_MASK_1
- EMEM_ADR_CFG_BANK_MASK_2
- EMEM_BANK_SWIZZLE_CFG0
- EMEM_BANK_SWIZZLE_CFG1
- EMEM_BANK_SWIZZLE_CFG2
- EMEM_BANK_SWIZZLE_CFG3

Offset: 0x664 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	ENABLED	EMEM_CFG_WRITE_ACCESS: 0 = ENABLED 1 = DISABLED

16.7.1.83 MC_TZ_SECURITY_CTRL_0

Trustzone Security Control Register

Controls "Strict"ness of TrustZone security check.

This register is "sticky" - once the strict check is enabled, it cannot be disabled without a reset.

Offset: 0x668 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	DISABLED	CPU_STRICT_TZ_APERTURE_CHECK: When this bit is set to ENABLED, the TrustZone security check requires that the CPU's security state bit exactly match the aperture. That is, when the CPU sets the NS bit to 0, it can ONLY access the TrustZone secured aperture. When the NS bit is 1, the CPU may NOT access the TrustZone secured aperture. Any violation causes a security violation to be flagged. When this bit is set to DISABLED, the security check is relaxed. In this mode, the CPU can access memory outside of the TrustZone secured aperture even though the NS bit is set to 0 (that is, even though the CPU is in secure mode). The above description only applies to the CPU complex clients (read and write). Other clients operate in the relaxed mode. 0 = DISABLED 1 = ENABLED

16.7.1.84 MC_EMEM_ARB_OUTSTANDING_REQ_RING3_0

External Memory Arbitration Configuration: Highest Ring Outstanding Request Limiter

> **Note:** Although called ring3, this register works on the highest ring.

This register is used to limit the number of outstanding requests in the arbiter and monitor the count. If the outstanding transactions are greater than the maximum, the highest ring requests are throttled based on the MC_EMEM_ARB_RING3_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 the Boot ROM section of warm boot, this register can be derived from other MC settings.

This register is shadowed: see usage note at the top of Section 16.7.1

Offset: 0x66c | Read/Write: R/W | Reset: 0x8XXX0080 (0b10xxxxxxxxxxxxxxxxxxxxx010000000)

Bit	R/W	Reset	Description
31	RW	ENABLE	LIMIT_OUTSTANDING_RING3: When ENABLED, requests into ring3 are throttled after the request count reaches ARB_MAX_OUTSTANDING_RING3. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	LIMIT_DURING_HOLDOFF_OVERRIDE_RING3: When DISABLED, overrides the limiting of ring3 transactions during holdoff to let requests flow freely. 0 = DISABLE 1 = ENABLE
24:16	RO	X	ARB_OUTSTANDING_COUNT_RING3: Current number of outstanding requests
8:0	RW	0x80	ARB_MAX_OUTSTANDING_RING3: The maximum number of requests allowed in the arbiter when LIMIT_OUTSTANDING is set to ENABLE

16.7.1.85 MC_SEC_CARVEOUT_BOM_0

Offset: 0x670 | Read/Write: R/W | Reset: 0xfff00000 (0b111111111111xxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:20	0xfff	SEC_CARVEOUT_BOM: [PMC_SECURE] Base address for the SEC carveout address space.

16.7.1.86 MC_SEC_CARVEOUT_SIZE_MB_0

Offset: 0x674 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	SEC_CARVEOUT_SIZE_MB: [PMC_SECURE] SEC_CARVEOUT_SIZE_MB is the size, in megabytes, of the SEC carveout region. If set to 0, the security check in MC is disabled.

16.7.1.87 MC_SEC_CARVEOUT_REG_CTRL_0

Offset: 0x678 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	ENABLED	SEC_CARVEOUT_WRITE_ACCESS: [PMC_SECURE] Sticky bit to control the writes to the other Sec Carveout aperture registers. 0 = ENABLED 1 = DISABLED

16.7.1.88 MC_ERR_SEC_STATUS_0

Offset: 0x67c | Read/Write: RO | Reset: 0x00XXX0XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
21:20	X	ERR_SEC_ADR_HI: Higher address bits of erring address whose lower bits are available in the ERR_ADR register
18	X	ERR_SEC_SWAP
17	X	ERR_SEC_SECURITY: Set if the transaction was secure. 0 = NONSECURE 1 = SECURE

Bit	Reset	Description
16	X	ERR_SEC_RW: Direction of the access that caused the error. 0 = READ 1 = WRITE
14:12	X	ERR_SEC_ADR1: Second subpartition unique address bits.
6:0	X	ERR_SEC_ID: Client identifier (see above list) of the access that caused the error.

16.7.1.89 MC_ERR_SEC_ADR_0

Offset: 0x680 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ERR_SEC_ADR

16.7.1.90 MC_PC_IDLE_CLOCK_GATE_CONFIG_0

Partition Idle Clock Gate Config

Boot requirements:

This register should be saved in the SDRAM and restored by the OS during warm boot.

Offset: 0x684 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx11111)

Bit	Reset	Description
4:0	0x1f	CLOCK_GATE_IDLE_TICKS: Number of idle "ticks" before a partition client is clock-gated

16.7.1.91 MC_STUTTER_CONTROL_0

Offset: 0x688 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	SLOW_EMCCLK_DURING_DSR: When ENABLED, the clock to the EMC will be slowed during dynamic self refresh. 0 = DISABLED 1 = ENABLED

16.7.1.92 MC_EMEM_ARB_NISO_THROTTLE_0

External Memory Arbitration Configuration: Snap Arbiter Throttle

Highest Ring Snap Arbiter Input Throttle - This register applies to the inputs of the highest ring. Refer to the EMEM_ARB_NISO_THROTTLE_MASK register to see how partition clients are configured into the "NISO meta-client" group for the purpose of NISO client throttling.

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 the top of Section 16.7.1

Offset: 0x6b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000xxxxxxxxxxx00000)

Bit	Reset	Description
20:16	0x0	NISO_THROTTLE_CYCLES_HIGH: Cycles of throttle after each request when the outstanding transaction count is greater than or equal to the threshold.

Bit	Reset	Description
4:0	0x0	NISO_THROTTLE_CYCLES: Cycles of throttle after each request.

16.7.1.93 MC_EMEM_ARB_OUTSTANDING_REQ_NISO_0

External Memory Arbitration Configuration: Highest Ring Input NISO Outstanding Request Limiter

Although called NISO, this register applies to the inputs of the highest snap-arbiter ring. Refer to the EMEM_ARB_NISO_THROTTLE_MASK register to see how partition clients are configured into the "NISO meta-client" group for the purpose of NISO client throttling.

This register can be used to limit the number of outstanding requests in the arbiter and monitor the count. If the outstanding transactions are greater than the maximum, the highest ring requests are throttled based on the MC_EMEM_ARB_NISO_THROTTLE 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 the Boot ROM section of warm boot, this register may be derived from other MC settings.

This register is shadowed: see usage note at the top of Section 16.7.1

Offset: 0x6b4 | Read/Write: R/W | Reset: 0x8XXX0080 (0b10xxxxxxxxxxxxxxxxxxxxx010000000)

Bit	R/W	Reset	Description
31	RW	ENABLE	LIMIT_OUTSTANDING_NISO: When ENABLED, requests into NISO are throttled after the request count reaches ARB_MAX_OUTSTANDING_NISO. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	LIMIT_DURING_HOLDOFF_OVERRIDE_NISO: When DISABLED, overrides the limiting of NISO transactions during holdoff to let requests flow freely. 0 = DISABLE 1 = ENABLE
24:16	RO	X	ARB_OUTSTANDING_COUNT_NISO: Current number of outstanding requests.
8:0	RW	0x80	ARB_MAX_OUTSTANDING_NISO: The maximum number of requests allowed in the arbiter when LIMIT_OUTSTANDING is set to ENABLE.

16.7.1.94 MC_EMEM_ARB_NISO_THROTTLE_MASK_0

External Memory Arbitration Configuration: Highest Ring Input NISO Throttle Mask

This register is used to group partition clients in the highest level snap-arbiter ring into a "NISO meta-client" group for the purpose of throttling. This allows soft-ISO clients in this ring to meet their bandwidth needs.

To prevent non-ISO (NISO) clients from filling the MC row-sorter (which could prevent soft-ISO clients from getting bandwidth), selective throttling is implemented at the input of ring2. To do this, the EMEM_ARB_NISO_THROTTLE_MASK register is used to specify which partition clients get included into a "NISO meta-client" group. To implement throttling, a total number of outstanding transactions counter (for example, only one counter exists in the whole "NV_MC_cif" to track this, and is shared by all throttling circuits) gets compared to a programmable maximum limit for this NISO meta-client group. If the number is greater than the maximum limit, all clients included in the "NISO meta-client" group get throttled based on the EMEM_ARB_NISO_THROTTLE register. This NISO_THROTTLE register is used to specify a number of stall cycles that get inserted at the input to the ring after every request from any of the clients included in the NISO group.

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0x6b8 | Read/Write: R/W | Reset: 0x00000000 (0bxx0000xx00xx0x0xx0xxxx000xxx000x)

Bit	Reset	Description
29	DISABLED	NISO_THROTTLE_MASK_VICPC: If enabled, include VICPC in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
28	DISABLED	NISO_THROTTLE_MASK_USBD: If enabled, include USBD in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
27	DISABLED	NISO_THROTTLE_MASK_HOST: If enabled, include HOST in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
26	DISABLED	NISO_THROTTLE_MASK_AUD: If enabled, include AUD in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
23	DISABLED	NISO_THROTTLE_MASK_SD: If enabled, include SD in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
22	DISABLED	NISO_THROTTLE_MASK_GK: If enabled, include GK in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
19	DISABLED	NISO_THROTTLE_MASK_MSE: If enabled, include MSE in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
17	DISABLED	NISO_THROTTLE_MASK_USBX: If enabled, include USBX in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
14	DISABLED	NISO_THROTTLE_MASK_VD: If enabled, include VD in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
9	DISABLED	NISO_THROTTLE_MASK_SAX: If enabled, include SAX in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
8	DISABLED	NISO_THROTTLE_MASK_FTOP: If enabled, include FTOP in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
7	DISABLED	NISO_THROTTLE_MASK_PCX: If enabled, include PCX in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
3	DISABLED	NISO_THROTTLE_MASK_AVP: If enabled, include AVP in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
2	DISABLED	NISO_THROTTLE_MASK_APB: If enabled, include APB in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
1	DISABLED	NISO_THROTTLE_MASK_AHB: If enabled, include AHB in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED

16.7.1.95 MC_EMEM_ARB_RING0_THROTTLE_MASK_0

External Memory Arbitration Configuration: Ring0 Input Throttle Mask

This register is used to group partition clients at the input to snaparb ring0 into a single "meta-client" group for throttling. This allows clients in this ring to meet their bandwidth needs.

Boot requirements:

- This register should be saved to SDRAM registers and restored by the OS during warm boot.
- This register is shadowed: see usage note at the top of Section 16.7.1

Offset: 0x6bc | Read/Write: R/W | Reset: 0x80008031 (0b1xxxxxxxxxxxxxxx1xxxxxxxx110001)

Bit	Reset	Description
31	ENABLE	ARB_OUTSTANDING_COUNT_ABOVE_SMMU: If enabled, arb_outstanding count includes requests in SMMU. If disabled, only count requests downstream of ring0. 0 = DISABLE 1 = ENABLE
15	ENABLE	RING0_THROTTLE_MASK_RING1_OUTPUT: If enabled, include the ring1 output (that is, all ring1 requests) in the ring0 meta-client group for throttling. 0 = DISABLE 1 = ENABLE
5	ENABLE	RING0_THROTTLE_MASK_R0_DISB: If enabled, include r0_disb in the ring0 meta-client group for throttling. 0 = DISABLE 1 = ENABLE
4	ENABLE	RING0_THROTTLE_MASK_R0_DIS: If enabled, include r0_dis in the ring0 meta-client group for throttling. 0 = DISABLE 1 = ENABLE
3	DISABLE	RING0_THROTTLE_MASK_MEM: If enabled, include mem in the ring0 meta-client group for throttling. 0 = DISABLE 1 = ENABLE
2	DISABLE	RING0_THROTTLE_MASK_PTC: If enabled, include ptc in the ring0 meta-client group for throttling. 0 = DISABLE 1 = ENABLE
1	DISABLE	RING0_THROTTLE_MASK_MMU: If enabled, include mmu in the ring0 meta-client group for throttling. 0 = DISABLE 1 = ENABLE
0	ENABLE	RING0_THROTTLE_MASK_MPCORER: If enabled, include mpcorer in the ring0 meta-client group for throttling. 0 = DISABLE 1 = ENABLE

16.7.1.96 MC_PC_IDLE_CLOCK_GATE_0

Partition Idle Clock Gate

Setting the control bit to NOT_GATED for a given partition will ensure that the mcclk, when the override is set to GATED_ON_IDLE, can be gated if that partition is idle.

Boot requirements:

This register should be saved in the SDRAM and restored by the OS during warm boot.

Offset: 0x954 | Read/Write: R/W | Reset: 0x3dfec29e (0bxx1111x11111111x11xxxx101xx1111x)

Bit	Reset	Description
29	GATED_ON_IDLE	VICPC_IDLE_CLOCK_GATE_ENABLE: Idle clock gate control for the vicpc partition 0 = NOT_GATED 1 = GATED_ON_IDLE
28	GATED_ON_IDLE	USBD_IDLE_CLOCK_GATE_ENABLE: Idle clock gate control for the usbd partition 0 = NOT_GATED 1 = GATED_ON_IDLE
27	GATED_ON_IDLE	HOST_IDLE_CLOCK_GATE_ENABLE: Idle clock gate control for the host partition 0 = NOT_GATED 1 = GATED_ON_IDLE
26	GATED_ON_IDLE	AUD_IDLE_CLOCK_GATE_ENABLE: Idle clock gate control for the aud partition 0 = NOT_GATED 1 = GATED_ON_IDLE
24	GATED_ON_IDLE	ISP_IDLE_CLOCK_GATE_ENABLE: Idle clock gate control for the isp partition 0 = NOT_GATED 1 = GATED_ON_IDLE
23	GATED_ON_IDLE	SD_IDLE_CLOCK_GATE_ENABLE: Idle clock gate control for the sd partition 0 = NOT_GATED 1 = GATED_ON_IDLE
22	GATED_ON_IDLE	GK_IDLE_CLOCK_GATE_ENABLE: Idle clock gate control for the gk partition 0 = NOT_GATED 1 = GATED_ON_IDLE
20	GATED_ON_IDLE	VE2_IDLE_CLOCK_GATE_ENABLE: Idle clock gate control for the ve2 partition 0 = NOT_GATED 1 = GATED_ON_IDLE
19	GATED_ON_IDLE	MSE_IDLE_CLOCK_GATE_ENABLE: Idle clock gate control for the mse partition 0 = NOT_GATED 1 = GATED_ON_IDLE
18	GATED_ON_IDLE	DISB_IDLE_CLOCK_GATE_ENABLE: Idle clock gate control for the disb partition 0 = NOT_GATED 1 = GATED_ON_IDLE
17	GATED_ON_IDLE	USBX_IDLE_CLOCK_GATE_ENABLE: Idle clock gate control for the usbx partition 0 = NOT_GATED 1 = GATED_ON_IDLE
15	GATED_ON_IDLE	VE_IDLE_CLOCK_GATE_ENABLE: Idle clock gate control for the ve partition 0 = NOT_GATED 1 = GATED_ON_IDLE
14	GATED_ON_IDLE	VD_IDLE_CLOCK_GATE_ENABLE: Idle clock gate control for the vd partition 0 = NOT_GATED 1 = GATED_ON_IDLE
9	GATED_ON_IDLE	SAX_IDLE_CLOCK_GATE_ENABLE: Idle clock gate control for the sax partition 0 = NOT_GATED 1 = GATED_ON_IDLE
8	NOT_GATED	FTOP_IDLE_CLOCK_GATE_ENABLE: Idle clock gate control for the ftop partition 0 = NOT_GATED 1 = GATED_ON_IDLE
7	GATED_ON_IDLE	PCX_IDLE_CLOCK_GATE_ENABLE: Idle clock gate control for the pcx partition 0 = NOT_GATED 1 = GATED_ON_IDLE
4	GATED_ON_IDLE	DIS_IDLE_CLOCK_GATE_ENABLE: Idle clock gate control for the dis partition 0 = NOT_GATED 1 = GATED_ON_IDLE

Bit	Reset	Description
3	GATED_ON_IDLE	AVP_IDLE_CLOCK_GATE_ENABLE: Idle clock gate control for the avp partition 0 = NOT_GATED 1 = GATED_ON_IDLE
2	GATED_ON_IDLE	APB_IDLE_CLOCK_GATE_ENABLE: Idle clock gate control for the apb partition 0 = NOT_GATED 1 = GATED_ON_IDLE
1	GATED_ON_IDLE	AHB_IDLE_CLOCK_GATE_ENABLE: Idle clock gate control for the ahb partition 0 = NOT_GATED 1 = GATED_ON_IDLE

16.7.1.97 MC_EMEM_ARB_OVERRIDE_1_0

Boot requirements:

- Some fields of register should be saved in the scratch registers and restored by the Boot ROM during warm boot.
- This security configuration register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0x968 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000x000000)

Bit	Reset	Description
31:7	0x0	EMEM_ARB_OVERRIDE_RESERVED
5	DISABLE	EXPIRE_UPDATE_DEADLOCK_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
4:2	DISABLE	CPU_READ_PAGE_OPEN_POLICY: 0 = DISABLE 1 = ENABLE_ALL 2 = ENABLE_EACK_ACTIVE 3 = ENABLE_EARLY_ACK_ENABLE 4 = ENABLE_PO_HINT
1:0	DISABLE	CPU_WRITE_PAGE_OPEN_POLICY : 0 = DISABLE 1 = ENABLE_ALL 2 = ENABLE_EACK_HINT

16.7.1.98 MC_CLIENT_HOTRESET_CTRL_1_0

Memory Client Hot Reset Control Register

Writing FLUSH_ENABLE to a bit in this register causes a flush to be performed on all 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:

- Set the appropriate FLUSH_ENABLE bit in the CLIENT_HOTRESET_CTRL register to block further access by the client, and start the flush process.
- Poll the CLIENT_HOTRESET_STATUS register until the appropriate bit returns FLUSH_DONE.
- Clear module bit in the CLK_RST_CONTROLLER_RST_DEVICES_* register to reset the module.
- Set module bit in the CLK_RST_CONTROLLER_RST_DEVICES_* register to release the module reset.
- Clear the appropriate FLUSH_ENABLE bit in the CLIENT_HOTRESET_CTRL register to allow transactions to flow.

Offset: 0x970 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	DISABLE	GPU_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
1	DISABLE	ISP2B_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
0	DISABLE	SDMMC4A_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED

16.7.1.99 MC_CLIENT_HOTRESET_STATUS_1_0

Memory Client Hot Reset Status Register

Contains one bit for each swname indicating the status of any flush that has been requested in the CLIENT_HOTRESET_CTRL register.

> **Note:** If no flush has been requested, this register returns FLUSH_DONE.

Offset: 0x974 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2	X	GPU_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
1	X	ISP2B_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
0	X	SDMMC4A_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS

16.7.1.100 MC_VIDEO_PROTECT_GPU_OVERRIDE_0_0

Offset: 0x984 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VIDEO_PROTECT_GPU_OVERRIDE_0

16.7.1.101 MC_VIDEO_PROTECT_GPU_OVERRIDE_1_0

Offset: 0x988 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	VIDEO_PROTECT_GPU_OVERRIDE_1

16.7.1.102 MC_MTS_CARVEOUT_BOM_0

Offset: 0x9a0 | Read/Write: R/W | Reset: 0xfff00000 (0b111111111111xxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:20	0xfff	MTS_CARVEOUT_BOM

16.7.1.103 MC_MTS_CARVEOUT_SIZE_MB_0

Offset: 0x9a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	MTS_CARVEOUT_SIZE_MB

16.7.1.104 MC_MTS_CARVEOUT_ADR_HI_0

Offset: 0x9a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	MTS_CARVEOUT_BOM_HI

16.7.1.105 MC_MTS_CARVEOUT_REG_CTRL_0

Offset: 0x9ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	ENABLED	MTS_CARVEOUT_WRITE_ACCESS 0 = ENABLED 1 = DISABLED

16.7.1.106 MC_SMMU_PTC_FLUSH_1_0

Program the higher address bits above PTC_FLUSH_ADR[31] in this field to flush a PDE or PTE group at a certain physical address.

Note: Only a 10 GB physical address is supported.

Page Table Cache Flush Address (Upper) Register

Offset: 0x9b8 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	PTC_FLUSH_ADR_HI: Physical address of PTE group to match for address flushes, PA[39:32]

16.7.1.107 MC_SECURITY_CFG3_0

Secure/Carveout Region Configuration: Base Address Higher Bits

Security Base:

- 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.

Offset: 0x9bc | Read/Write: R/W | Secure Trust Zone Protected | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	SECURITY_BOM_HI: SECURITY_BOM_HI has the higher address bits beyond 32 bits of the base of the secured region, limited to MB granularity.

16.7.1.108 MC_EMEM_BANK_SWIZZLE_CFG0_0

Bank Swizzling Register

Offset: 0x9c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	EMEM_BANK_MASK0

16.7.1.109 MC_EMEM_BANK_SWIZZLE_CFG1_0

Bank Swizzling Register

Offset: 0x9c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	EMEM_BANK_MASK1

16.7.1.110 MC_EMEM_BANK_SWIZZLE_CFG2_0

Bank Swizzling Register

Offset: 0x9c8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	EMEM_BANK_MASK2

16.7.1.111 MC_EMEM_BANK_SWIZZLE_CFG3_0

Bank Swizzling Register

Offset: 0x9cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2:0	COL_NONE	EMEM_BANK_SWIZCOL 0 = COL_NONE 1 = COL_1KB 2 = COL_2KB 3 = COL_2KB_OFFSET_2 4 = COL_2KB_OFFSET_4

16.7.1.112 MC_SEC_CARVEOUT_ADR_HI_0

Offset: 0x9d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	SEC_CARVEOUT_BOM_HI: [PMC_SECURE] Higher address bits beyond 32 bits of byte-aligned address of the base address of the SEC carveout space.

16.7.1.113 MC_SMMU_DC1_ASID_0

SMMU DC1 ASID Assignment Register

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).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xa88 | Read/Write: R/W | Secure Trust Zone Protected | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	DC1_SMMU_ENABLE: If set, translation is enabled for this client. 0 = DISABLE 1 = ENABLE
6:0	0x0	DC1_ASID: ASID to use for translation, if enabled.

16.7.1.114 MC_SMMU_SDMMC1A_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).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

SMMU SDMMC1A ASID Assignment Register

Offset: 0xa94 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	SDMMC1A_SMMU_ENABLE: If set, translation is enabled for this client. 0 = DISABLE 1 = ENABLE
6:0	0x0	SDMMC1A_ASID: ASID to use for translation, if enabled.

16.7.1.115 MC_SMMU_SDMMC2A_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).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

SMMU SDMMC2A ASID Assignment Register

Offset: 0xa98 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	SDMMC2A_SMMU_ENABLE: If set, translation is enabled for this client. 0 = DISABLE 1 = ENABLE
6:0	0x0	SDMMC2A_ASID: ASID to use for translation, if enabled.

16.7.1.116 MC_SMMU_SDMMC3A_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).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

SMMU SDMMC3A ASID Assignment Register

Offset: 0xa9c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	SDMMC3A_SMMU_ENABLE: If set, translation is enabled for this client. 0 = DISABLE 1 = ENABLE
6:0	0x0	SDMMC3A_ASID: ASID to use for translation, if enabled.

16.7.1.117 MC_SMMU_SDMMC4A_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).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

SMMU SDMMC4A ASID Assignment Register

Offset: 0xaa0 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	SDMMC4A_SMMU_ENABLE: If set, translation is enabled for this client. 0 = DISABLE 1 = ENABLE
6:0	0x0	SDMMC4A_ASID: ASID to use for translation, if enabled.

16.7.1.118 MC_SMMU_ISP2B_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).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

SMMU ISP2B ASID Assignment Register

Offset: 0xaa4 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	ISP2B_SMMU_ENABLE: If set, translation is enabled for this client. 0 = DISABLE 1 = ENABLE
6:0	0x0	ISP2B_ASID: ASID to use for translation, if enabled.

16.7.1.119 MC_SMMU_GPU_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).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

SMMU GPU ASID Assignment Register

Offset: 0xaa8 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	GPU_SMMU_ENABLE: Hardcoded for GPU to DISABLE. 0 = DISABLE 1 = ENABLE
6:0	0x0	GPU_ASID: ASID to use for translation, if enabled.

16.7.1.120 MC_SMMU_GPUB_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).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

SMMU GPUB ASID Assignment Register

Offset: 0xaac | Read/Write: R/W | Reset: 0x80000000 (0b1xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	ENABLE	GPUB_SMMU_ENABLE: Hardcoded for GPUB to ENABLE. 0 = DISABLE 1 = ENABLE
6:0	0x0	GPUB_ASID: ASID to use for translation, if enabled.

16.7.1.121 MC_SMMU_PPCS2_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).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

SMMU PPCS2 ASID Assignment Register

Offset: 0xab0 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	PPCS2_SMMU_ENABLE: If set, translation is enabled for this client. 0 = DISABLE 1 = ENABLE
6:0	0x0	PPCS2_ASID: ASID to use for translation, if enabled.

16.7.2 EMC Registers

USAGE NOTE: Many EMC register fields are shadowed:

- Writes to shadowed register fields update the shadow copy (this is default, assumes DBG.WRITE_MUX==ASSEMBLY).
- Reads to shadowed register fields return the currently-active copy (this is default, assumes DBG.READ_MUX==ACTIVE).

This allows a new set of frequency-dependent timing parameters to be written to the shadow registers while memory traffic is ongoing, then when the parameters are completely written, the EMC hardware can perform a timing-safe switch.

Such switches can be triggered via two methods: TIMING_CONTROL.TIMING_UPDATE (generally used during initialization), or the automatic CAR/EMC handshake on a clock-frequency or divider change (assuming CFG_2.CLKCHANGE_REQ_ENABLE==ENABLED).

Registers that are shadowed are marked by a comment:

This register is shadowed: see usage note at the top of Section 16.7.2

Occasionally, only certain fields in the register will be shadowed; if so they are noted after the above comment.

USAGE NOTE: Many EMC registers play crucial roles in warm boot (also known as "wake from LP0") and cold boot (also known as "power up") sequences. Suggested actions for the Boot ROM/Boot Loader are noted after "Boot requirements:".

USAGE NOTE: The EMC register fields to be stored in the PMC must have "[PMC]", "[PMC2]", or "[PMC3]".in their comments. ([PMC2] registers are packed/unpacked after [PMC] register group in software code. [PMC3] is even later.) The fields that need to be stored in PMC Secure Scratch registers must have [PMC_SECURE] in their comments.

USAGE NOTE: When writing to a register, any bits that are not intended to be modified should be rewritten with their existing values.

USAGE NOTE: Unspecified bits may not appear in tables and should be written with their Reset values.

16.7.2.1 EMC_INTSTATUS_0

Interrupt Status Register

Clear on 1-write. Init value is clear. This register is “sticky”.

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000xxx)

Bit	Reset	Description
9	CLEAR	DLL_LOCK_TIMEOUT_INT: Indicates a DLL lock timeout has occurred. 0 = CLEAR 1 = SET
8	CLEAR	CCFIFO_OVERFLOW_INT: Clock change FIFO overflow. 0 = CLEAR 1 = SET
7	CLEAR	DLL_ALARM_INT: Indicates a DLL alarm has been set 0 = CLEAR 1 = SET
6	CLEAR	ACCESS_TO_SR_DPD_DEV_INT: Indicates a system attempt to access a self-refresh/deep-powered-down device. 0 = CLEAR 1 = SET
5	CLEAR	MRR_DIVLD_INT: LPDDR3 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

16.7.2.2 EMC_INTMASK_0

Interrupt Mask Register

Init value is masked.

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000xxx)

Bit	Reset	Description
9	MASKED	DLL_LOCK_TIMEOUT_INTMASK: Mask for DLL lock timeout interrupt. 0 = MASKED 1 = UNMASKED
8	MASKED	CCFIFO_OVERFLOW_INTMASK: Mask for clock change FIFO overflow. 0 = MASKED 1 = UNMASKED

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

16.7.2.3 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.

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0xc | Read/Write: R/W | Reset: 0x03c0000e (0b0000x01111000000xxxxxx000000111x)

Bit	Reset	Description
31	DISABLED	DRAM_CLKSTOP_PD: [PMC] clock stop is only allowed to happen if the DRAM is in active/precharge powerdown (for all CKE bits associated with clock) 0 = DISABLED 1 = ENABLED
30	DISABLED	DRAM_CLKSTOP_SR: [PMC] clockstop is only allowed to happen if the DRAM is in self-refresh (for all CKE bits associated with clock) 0 = DISABLED 1 = ENABLED
29	NO_POWERDOWN	DRAM_ACPD: [PMC] Allows the DRAM controller to perform opportunistic active powerdown control using the CKE pin on the DRAM. The behavior of the powerdown control logic is controlled by the PDEX2* and *2PDEN registers. The value of DRAM_ACPD should only be changed when CKE is low, e.g., during software-controlled self-refresh or before DRAM initialization. 0 = NO_POWERDOWN 1 = ACTIVE_POWERDOWN
28	DISABLED	DYN_SELF_REF: [PMC] 0 = DISABLED 1 = ENABLED
26	DISABLED	REQACT_ASYNC: [PMC] Allows independent transfers of activates and transactions from the MC to the EMC. Disabled to preserve the order of requests selected by mc_arb logic. 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
25	ENABLED	AUTO_PRE_WR: [PMC] enable using MC auto-precharge indication for writes. This bits, when set to DISABLE, will override the settings in the MC register (EMC ignore the auto-precharge indication from MC). Otherwise, they permit clients to make either auto or explicit precharge requests depend on the state of MAN_PRE_WR. 0 = DISABLED 1 = ENABLED
24	ENABLED	AUTO_PRE_RD: [PMC] enable using MC auto-precharge indication for reads. This bits, when set to DISABLE, will override the settings in the MC register (EMC ignores the auto-precharge indication from MC). Otherwise, they permit clients to make either auto or explicit precharge requests depend on the state of MAN_PRE_RD. 0 = DISABLED 1 = ENABLED
23	ENABLED	MAN_PRE_WR: [PMC] enable explicit-precharge in the EMC for writes. When this bit is enabled (and AUTO_PRE_WR=enable), EMC will issue an explicit precharge command after the memory write command. If this bit is disabled (and AUTO_PRE_WR=enable), the EMC will issue an auto-precharge with the memory write command. 0 = DISABLED 1 = ENABLED
22	ENABLED	MAN_PRE_RD: [PMC] enable explicit-precharge in the EMC for reads. When this bit is enabled (and AUTO_PRE_RD=enable), EMC will issue an explicit precharge command after the memory read command. If this bit is disabled (and AUTO_PRE_RD=enable), the EMC will issue an auto-precharge with the memory read command. 0 = DISABLED 1 = ENABLED
21	DISABLED	PERIODIC_QRST: [PMC] specifies whether or not to periodic reset the FBIO read-data FIFO during normal operation. The periodic resets can be used for graceful recovery from an intermittent failure condition; only the initial reset is absolutely required. Note: For LPDDRx and MRR (if ever used), this bit should be set to enable. 0 = DISABLED 1 = ENABLED
20	DISABLED	EN_DYNAMIC_PUTERM: [PMC] enable dynamic puterm. 0 = DISABLED 1 = ENABLED
19	DISABLED	DLY_WR_DQ_HALF_CLOCK: [PMC] 1 = delay write data by 1/2 clock. 0 = DISABLED 1 = ENABLED
18	DISABLED	DSR_VTTGEN_DRV_EN: [PMC] enable VTTGEN controls (via DSR_VTTGEN_DRV/TXDSRVTTGEN registers) during DSR. 0 = DISABLED 1 = ENABLED
17:16	TWOX	EMC2MC_CLK_RATIO: [PMC] EMC to MC clock ratio. 0 = TWOX 1 = ONEX 2 = FOURX 3 = RESERVED
9	0x0	WAIT_FOR_ISP2B_READY_B4_CC: [PMC] 1 = wait for isp2b ready event to be asserted before acknowledge clock change.
8	0x0	WAIT_FOR_VI2_READY_B4_CC: [PMC] 1 = wait for vi2 ready event to be asserted before acknowledge clock change.
7	0x0	WAIT_FOR_ISP2_READY_B4_CC: [PMC] 1 = wait for isp2 ready event to be asserted before acknowledge clock change.
6	0x0	INVERT_DQM: [PMC] 1 = invert DQM polarity.

Bit	Reset	Description
5	0x0	WAIT_FOR_DISPLAYB_READY_B4_CC: [PMC] 1 = wait for displayb ready event to be asserted before acknowledging clock change.
4	0x0	WAIT_FOR_DISPLAY_READY_B4_CC: [PMC] 1 = wait for display ready event to be asserted before acknowledging clock change.
3	0x1	EMC2PMACRO_CFG_BYPASS_DATAPIPE2: [PMC] 1 = bypass data pipeline stage2 (closer to I/O) between EMC and data pad-macro.
2	0x1	EMC2PMACRO_CFG_BYPASS_DATAPIPE1: [PMC] 1 = bypass data pipeline stage1 (closer to core) between EMC and data pad-macro.
1	0x1	EMC2PMACRO_CFG_BYPASS_ADDRPIPE: [PMC] 1 = bypass address pipeline stage between EMC and address pad-macro.

16.7.2.4 EMC_ADR_CFG_0

External Memory Address Configuration, System

The ADR_CFG register is used to specify the number of DRAM devices. DRAM density and geometry parameters are specified by the EMEM_ADR_CFG* registers in MC.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	N2	EMEM_NUMDEV: [PMC SECURE] Number of populated DRAM devices. 0 = N1 1 = N2

16.7.2.5 EMC_REFCTRL_0

Refresh Control Register

The REFCTRL register allows software to enable or disable the refresh controller.

REF_VALID should be enabled after the initialization sequence is completed.

Boot requirements:

- If per-device DPD is used, the DEVICE_REFRESH_DISABLE field should be parameterized in the BCT and written by the Boot ROM during cold boot.
- If per-device DPD is used, the DEVICE_REFRESH_DISABLE field should be parameterized the scratch registers and restored by the Boot ROM during warm boot.
- REF_VALID field should always be set to ENABLED for normal use, no need to parameterize in BCT/scratch.

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
31	DISABLED	REF_VALID: Enable refresh controller. 0 = DISABLED 1 = ENABLED
1:0	0x0	DEVICE_REFRESH_DISABLE: Disables refresh to individual attached device (1 bit per DRAM chip select).

16.7.2.6 EMC_PIN_0

Controls State of Selected DRAM Pins

The PIN register allows software to control the state of the selected external DRAM pins.

Boot requirements:

- Both fields in this register should be set to NORMAL during cold boot.
- Both fields in this register should be set to NORMAL during warm boot.

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000100 (0bxxxxxxxxxxxxxxxxxxxxxxxx1xxx0xxx0)

Bit	Reset	Description
8	INACTIVE	PIN_RESET: Selects the level of the DDR3 RESET# pin. 0 = ACTIVE 1 = INACTIVE
4	NORMAL	PIN_DQM: Is used to always mask DRAM writes. This pin should only be used for initialization. Certain DRAM vendors require the DQM to be high during initialization. The register value should be set to NORMAL after the initialization sequence. 0 = NORMAL 1 = INACTIVE
0	POWERDOWN	PIN_CKE: Selects the level of the CKE pin. This can be used to place the DRAM in power down state. PIN_CKE value is applied all CKE pins. 0 = POWERDOWN 1 = NORMAL

16.7.2.7 EMC_TIMING_CONTROL_0

Triggers an Update of the Timing-Related Registers

The TIMING_CONTROL register is used by software to trigger parameter updates for timing parameter registers, RDQS/QUSE delay controls, and some DLL controls. Writing the TIMING_UPDATE field updates the active state of these registers with the programmed assembly state. The active state is updated during a safe interval determined by the EMC. If CLKCHANGE_REQ_ENABLE is enabled, the active value will automatically be updated on completion of the clock change.

> **Note:** Programming of this register does not trigger the shadow register update event immediately. To prevent shadow register programming issued after programming this register from being latched accidentally, always poll for TIMING_UPDATE_STALLED==0 after programming this register.

Boot requirements:

- Writing this register with 0x1 will trigger a timing update event, which should be used in both warm boot and cold boot sequences.

Offset: 0x28 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	TIMING_UPDATE

16.7.2.8 EMC_RC_0

DRAM Timing Parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2c | Read/Write: R/W | Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111)

Bit	Reset	Description
6:0	0x7f	RC: [PMC] Specifies the row cycle time. This is the minimum number of cycles between activate commands to the same bank. LPDDR3 : ceil((tRASmin+tRPpb)/tCK) – 1 DDR3: max(0, ceil(tRC/tCK) - (CMD_2T_TIMING == 1 ? 2 : 1))

16.7.2.9 EMC_RFC_0

DRAM Timing Parameter

This register is shadowed: see usage note at the top of Section 16.7.2 .

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x30 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxx000111111)

Bit	Reset	Description
8:0	0x3f	RFC: [PMC] specifies the auto refresh cycle time. This is the minimum number of cycles between an auto refresh command and a subsequent auto refresh or activate command. LPDDR3: ceil(tRFCab/tCK) – 1 DDR2 : ceil(tRFC/tCK) - 1 DDR3: max(0, ceil(tRFC/tCK) - (CMD_2T_TIMING == 1 ? 2 : 1))

16.7.2.10 EMC_RAS_0

DRAM Timing Parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x34 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	RAS: [PMC] specifies the row active time. This is the minimum number of cycles between an activate command and a precharge command to the same bank. DDR3: max(0, ceil(tRASmin/tCK) - (CMD_2T_TIMING == 1 ? 2 : 1)) LPDDR3: max(3, ceil(tRASmin/tCK)) - 1

16.7.2.11 EMC_RP_0

DRAM Timing Parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x38 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	RP: [PMC] specifies the row precharge time. This is the minimum number of cycles between a precharge command and an activate command to the same bank. DDR3: max(0, ceil(tRP/tCK) - (CMD_2T_TIMING == 1 ? 2 : 1)) LPDDR3: max(3, ceil(tRP/tCK)) - 1

16.7.2.12 EMC_R2W_0

R2something and W2something registers are independent of burst length, and are relative to the last virtual CAS_ of a command. Counting starts from the last data transfer as related to where CAS_ would be if BL = 4.

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x3c | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx11111)

Bit	Reset	Description
4:0	0x1f	R2W: [PMC] specifies the minimum number of cycles from any read command to any write command, irrespective of bank. This parameter guarantees the read->write turnaround 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) LPDDR3: 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

16.7.2.13 EMC_W2R_0

R2something and W2something registers are independent of burst length, and are relative to the last virtual CAS_ of a command. Counting starts from the last data transfer as related to where CAS_ would be if BL = 4.

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x40 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx11111)

Bit	Reset	Description
4:0	0x1f	W2R: [PMC] specifies the minimum number of cycles from a write command to a read command, irrespective of bank. DDR3 : WL + max(4, ceil(tWTR/tCK)) - (CMD_2T_TIMING == 1 ? 1 : 0) LPDDR3 : WL + 1 + ceil(tWTR/tCK) Largest programming value is 29.

16.7.2.14 EMC_R2P_0

R2something and W2something registers are independent of burst length, and are relative to the last virtual CAS_ of a command. Counting starts from the last data transfer as related to where CAS_ would be if BL = 4.

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x44 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx11111)

Bit	Reset	Description
4:0	0x1f	R2P: [PMC] specifies the minimum number of cycles from a read command to a precharge command for the same bank. DDR3 : max(4, ceil(tRTP/tCK)) - (CMD_2T_TIMING == 1 ? 2 : 1) LPDDR3: BL/2 + max(2, ceil(tRTP/tCK)) 4 .

16.7.2.15 EMC_W2P_0

R2something and W2something registers are independent of burst length, and are relative to the last virtual CAS_ of a command. Counting starts from the last data transfer as related to where CAS_ would be if BL = 4.

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x48 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	W2P: [PMC] Specifies the minimum number of cycles from a write command to a precharge command for the same bank. DDR3 : WL + max(2, ceil(tWR/tCK)) - (CMD_2T_TIMING == 1 ? 1 : 0) LPDDR3: WL + max(3, TWR)

16.7.2.16 EMC_RD_RCD_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x4c | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx011111)

Bit	Reset	Description
5:0	0x1f	RD_RCD: [PMC] Specifies the RAS to CAS delay. RD_RCD is the minimum number of cycles between an activate command and a read command to the same bank. DDR3 : ceil(tRCD/tCK) - (CMD_2T_TIMING == 1 ? 2 : 1) LPDDR3: max(3, ceil(tRCD/tCK)) - 1

16.7.2.17 EMC_WR_RCD_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x50 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx011111)

Bit	Reset	Description
5:0	0x1f	WR_RCD: [PMC] Minimum number of cycles between an activate command and a write command to the same bank. DDR3 : ceil(tRCD/tCK) - (CMD_2T_TIMING == 1 ? 2 : 1) LPDDR3: max(3, ceil(tRCD/tCK)) - 1

16.7.2.18 EMC_RRD_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x54 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111)

Bit	Reset	Description
3:0	0xf	RRD: [PMC] specifies the Bank X Act to Bank Y Act command delay. DDR3 : max(4, ceil(tRRD/tCK)) - (CMD_2T_TIMING == 1 ? 2 : 1) LPDDR3: max(2, ceil(tRRD/tCK)) - 1

16.7.2.19 EMC_REXT_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0001)

Bit	Reset	Description
3:0	0x1	REXT: [PMC] specifies the read to read delay for reads when multiple physical devices are present. DDR3: USE_PER_DEVICE_DLY_TRIM_IB == 1 ? 2 : 1 LPDDR3: ceil(3/tCK + 0.5) + USE_PER_DEVICE_DLY_TRIM_IB ? 1 : 0

16.7.2.20 EMC_WDV_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	WDV: [PMC] The number of cycles to post (delay) write data from being asserted to the RAMs. DDR3 : WL - 1 LPDDR3 : WL 15 = MAX

16.7.2.21 EMC_QUSE_0

DRAM timing parameter

The QRST, QUSE, and RDV fields specify the delays from read to internal timing signals. These fields should be set as follows:

QUSE = CAS_LATENCY - 1 non-mobile SDRAM
= CAS_LATENCY - 2 for mobile SDRAM
QRST = CAS_LATENCY - 2
QSAFE >= RDV - QRST
RDV = CAS_LATENCY + 5
EINPUT = RDV - 7

Because SDRAM uses a tristating clock (the DQS), a method is needed to deal with the ambiguity of when DQS is tristated. Only one such method is supported: QUSE.

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x60 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000010)

Bit	Reset	Description
5:0	0x2	QUSE: [PMC] Tells the chip when to look for read return data. LPDDR3/DDR3: obtained from characterization

16.7.2.22 EMC_QRST_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x64 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000001)

Bit	Reset	Description
5:0	0x1	QRST: [PMC] time from expiration of QSAFE until reset is issued. DDR3 : CL - 2 LPDDR3: RL - 2

16.7.2.23 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 (for example, the queue is empty). This field must be set to at least (RDV - QRST).

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x68 | Read/Write: R/W | Reset: 0x00000007 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00111)

Bit	Reset	Description
4:0	0x7	QSAFE: [PMC] Time from a read command to when it is safe to issue a QRST (delayed by the QRST parameter). LPDDR3/DDR3 QSAFE >= RDV - QRST

16.7.2.24 EMC_RDV_0

DRAM timing parameter

RDV is the read latency register. This register value is not negotiable; it will work with the right value and will not with the wrong value. It uses sequential timing, not combinational (i.e., maybe the silicon is fast enough type) timing.

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x6c | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx001000)

Bit	Reset	Description
5:0	0x8	RDV: [PMC] Time from read command to latching the read data from the pad macros. DDR3 : RL + 6 + f_ceil(fly-by-time/tCK) - (dram-dll-is-off ? 1 : 0) LPDDR3: RL + 6 + ceil((tDQSCKmax + fly-by-time)/tCK) Fly-by-time is board routing dependent, which usually should be less than 1.5ns. dram-dll-is-off is true below 150 MHz where DRAM DLL has to be turned off 45 = MAX

16.7.2.25 EMC_REFRESH_0

DRAM timing parameter

The value of REFRESH is calculated using the following formula:

```
{REFRESH, REFRESH_LO} = max[(tREF/#_of_rows) / (emc_clk_period) - 64,
                            (tREF/#_of_rows) / (emc_clk_period) * 97%]
```

For example, if the clock frequency is 133 MHz, and the refresh requirement is 64 ms per 4096 rows. The programming value is 0x7e3.

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x70 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxx0000000000011111)

Bit	Reset	Description
15:6	0x0	REFRESH: [PMC] Specifies the interval between refresh requests.
5:0	0x1f	REFRESH_LO: [PMC]

16.7.2.26 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 the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x74 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	BR1	BURST_REFRESH_NUM: [PMC] Specify the refresh burst count. 0 = BR1 1 = BR2 2 = BR4 3 = BR8 4 = BR16 5 = BR32 6 = BR64 7 = BR128 8 = BR256 9 = BR512 9 = MAX

16.7.2.27 EMC_PDEX2WR_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x78 | Read/Write: R/W | Reset: 0x0000003e (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111110)

Bit	Reset	Description
5:0	0x3e	PDEX2WR: Specify the timing delay from exit of powerdown mode to a write command. DDR3 : max(3, ceil(tXP/tCK)) - (CMD_2T_TIMING == 1 ? 2 : 1) LPDDR3 : 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. The largest allowed value is 62.

16.7.2.28 EMC_PDEX2RD_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x7c | Read/Write: R/W | Reset: 0x0000003e (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111110)

Bit	Reset	Description
5:0	0x3e	PDEX2RD: Specify the timing delay from exit of powerdown mode to a read command. DDR3: (use-slow-pd-exit-mode ? max(10, ceil(tXPDLL/tCK)) : max(3, ceil(tXP/tCK))) - (CMD_2T_TIMING == 1 ? 2 : 1) LPDDR3: ceil(tXP/tCK) - 1 use-slow-pd-exit-mode is 1 if slow power down exit mode is used (MR0 bit 12 is 0). The largest allowed value is 62.

16.7.2.29 EMC_PCHG2PDEN_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x80 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx011111)

Bit	Reset	Description
5:0	0x1f	PCHG2PDEN: [PMC] Specify the timing delay from a precharge command to powerdown entry. DDR3: 1 LPDDR3: f_ceil(tRP]/tCK) - (CMD_2T_TIMING == 1 ? 1 : 0)

16.7.2.30 EMC_ACT2PDEN_0

[PMC] DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x84 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx011111)

Bit	Reset	Description
5:0	0x1f	ACT2PDEN: Specify the timing delay from an activate, MRS or EMRS command to power-down entry. DDR3/ LPDDR3: 0

16.7.2.31 EMC_AR2PDEN_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x88 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxx000011111)

Bit	Reset	Description
8:0	0x1f	AR2PDEN: [PMC] Specify the timing delay from an autorefresh command to powerdown entry. DDR3: max(7, ceil((tXPDLL - tXP)/tCK)) LPDDR3: 1

16.7.2.32 EMC_RW2PDEN_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x8c | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx011111)

Bit	Reset	Description
5:0	0x1f	RW2PDEN: [PMC] Specify the timing delay from a read/write command to powerdown entry. Auto-precharge timing must be taken into account when programming this field DDR3: max(RL+5, WL+4+TWR) - 1 LPDDR3: max(RL+BL/2+ceil((tDQSCKmax+1)/tCK)+1, WL+BL/2+1+ceil((tWR+1)/tCCK)) - 1

16.7.2.33 EMC_TXSR_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x90 | Read/Write: R/W | Reset: 0x000003fe (0bxxxxxxxxxxxxxxxxxxxxxxxx1111111110)

Bit	Reset	Description
9:0	0x3fe	TXSR: [PMC] Cycles between self-refresh exit & first DRAM command that does not require a locked DLL. Largest allowed value is 0x3fe. DDR3: max(5, ceil(tXSR/tCK)) LPDDR3: max(2, ceil(tXSR/tCK))

16.7.2.34 EMC_TCKE_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x94 | Read/Write: R/W | Reset: 0x0000003e (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111110)

Bit	Reset	Description
5:0	0x3e	TCKE: Specify the minimum CKE high pulse width. DDR3: Use-slow-pd-exit-mode ? max(10, ceil(max(tCKESR, uXPDLL)/tCK)) : max(4, ceil(max(tXP,tXP)/tCK)) LPDDR3: 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).

16.7.2.35 EMC_TFAW_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x98 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	TFAW: [PMC] specify the width of the FAW (four-activate window) for 8-bank devices. Set to 0 to disable this timing check. Only 4 activates may occur within the rolling window. . DDR3/LPDDR3: ceil(tFAW/tCK)

16.7.2.36 EMC_TRPAB_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x9c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

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 LPDDR3: max(3, ceil((tRPpb+3)/tCK

16.7.2.37 EMC_TCLKSTABLE_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xa0 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxx01111)

Bit	Reset	Description
4:0	0xf	TCLKSTABLE: [PMC] Specify minimum number of cycles of a stable clock period prior to exiting powerdown or self-refresh modes. DDR3: max(5, ceil(tCKSRX/tCK)) - 1 LPDDR3: 2

16.7.2.38 EMC_TCLKSTOP_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xa4 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx01111)

Bit	Reset	Description
4:0	0xf	TCLKSTOP: [PMC] Delay from last command to stopping the external clock to DRAM devices. DDR3: max(5, ceil(tCKSRE/tCK)) - 1 LPDDR3: 2

16.7.2.39 EMC_TREFBW_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xa8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00000000000000)

Bit	Reset	Description
13:0	0x0	TREFBW: [PMC] specify the width of the burst-refresh window. If set to a non-zero value, only 8 refreshes will occur in this rolling window. Set to 0 to disable this timing check. DDR3: ceil(tREFI/tCK) LPDDR3: ceil(tREFBW/tCK)

16.7.2.40 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 the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xb0 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxx0000xx000000)

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 the first 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.

Bit	Reset	Description
11:8	0x0	ODT_WR_DURATION: [PMC] for LPDDR3, indicate how long to assert ODT by.
5	0x0	SHARE_ONE_ODT: [PMC] For LPDDR3 only. When enabled, ODT[1] is not used. The ODT for both ranks will be sharing the same ODT[0]. When disabled, ODT[0] is controlling rank0, ODT[1] is controlling rank1.
4	0x0	DRIVE_BOTH_ODT: [PMC] If this field = 0, only the ODT for the requested device will be asserted during write. If this field = 1, ODTs for both devices will be asserted during write.
3:0	0x0	ODT_WR_DELAY: [PMC] Set this field = ABS (WL - ceiling(tAOND) - 2). The valid programming range is: 0 <= ODT_WR_DELAY <= 2, if ODT_B4_WRITE=0 0 <= ODT_WR_DELAY <= 1, if ODT_B4_WRITE=1. For DDR3, this field = (WL - 3). For LPDDR3, this field = (WL - 2).

16.7.2.41 EMC_ODT_READ_0

This register is not used. Please keep its reset value.

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xb4 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
31	0x0	DISABLE_ODT_DURING_READ: enables ODT to be turned off prior to issuing read to DRAM. If this field == 0, the ODT state is not 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 a 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.

16.7.2.42 EMC_WEXT_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xb8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

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)) LPDDR3: USE_PER_DEVICE_DLY_TRIM_OB == 1 ? 5 : 0

16.7.2.43 EMC_CTT_0

DRAM timing parameter

CTT controls timing of internal read terminations. Based on read latency (CTT must frame read data). Adding non-RDV linked timing control for flexibility. Has no effect unless CTT_TERMINATION == ENABLED || E_PUTERM_DQ == ENABLED It is sequential timing, not combinational (i.e., maybe the silicon is fast enough type) timing.

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0xbc | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx001000)

Bit	Reset	Description
5:0	0x8	CTT: [PMC] time from read command to CTT turn-on. DDR3 ? (CTT_TERMINATION == 1) ? CL + 7 : 0 LPDDR3: 0 45 = MAX

16.7.2.44 EMC_RFC_SLR_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8:0	0x0	RFC_SLR: [PMC] Specifies the stagger refresh cycle time between ranks. This is the minimum number of cycles between an auto refresh command to one rank and a subsequent auto refresh to another rank. A zero value indicates stagger refresh is disabled.

16.7.2.45 EMC_MRS_WAIT_CNT2_0

This register allows the delay between an MRS/EMRS/MRW/MRR command and the following DRAM command

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0xc4 | Read/Write: R/W | Reset: 0x0208000f (0bxxxxxx1000001000xxxxxx0000001111) | Default: 0x03ff03ff

Bit	Reset	SW Default	Description
25:16	0x208	0x3ff	MRS_EXT2_WAIT_CNT: [PMC] Number of EMC clocks to wait before issuing

Bit	Reset	SW Default	Description
			any commands after sending an MRS ext2 command.
9:0	0xf	0x3ff	MRS_EXT1_WAIT_CNT: [PMC] Number of EMC clocks to wait before issuing any commands after sending an MRS ext1 command.

16.7.2.46 EMC_MRS_WAIT_CNT_0

This register allows the delay between an MRS/EMRS/MRW/MRR command and the following DRAM command.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0xc8 | Read/Write: R/W | Reset: 0x0208000f (0bxxxxxx1000001000xxxxxx0000001111)

Bit	Reset	SW Default	Description
25:16	0x208	0x3ff	MRS_LONG_WAIT_CNT: [PMC] Number of EMC clocks to wait before issuing any commands after sending an MRS long command.
9:0	0xf	0x3ff	MRS_SHORT_WAIT_CNT: [PMC] Number of EMC clocks to wait before issuing any commands after sending an MRS short command.

16.7.2.47 EMC_MRS_0

Command Trigger: MRS

The MRS register allows software to issue an MRS command.

BA0, BA1 are used to address MRS or EMRS registers in DRAM. Although this register can also program EMRS, use the EMRS register so that the hardware registers can shadow what is in the DRAM.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0xcc | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xxxx00xxxxxx00000000000000)

Bit	Reset	Description
31:30	0x0	MRS_DEV_SELECTN: [PMC] Active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
27	0x0	USE_MRS_EXT_CNT: [PMC] 0 =USE_MRS_LONG_CNT will select between SHORT(0)/LONG(1), 1=USE_MRS_LONG_CNT will select between EXT1(0)/EXT2(1).
26	SHORT	USE_MRS_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
21:20	0x0	MRS_BA: [PMC] Set to 0x0 for MRS.
13:0	0x0	MRS_ADR: [PMC] Mode-register data to be written.

16.7.2.48 EMC_EMRS_0

Command trigger: EMRS

The EMRS register allows software to issue an EMRS command.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0xd0 | Read/Write: R/W | Reset: 0x00000000 (0b00xxx0xxxx00xxxxxx00000000000000)

Bit	Reset	Description
31:30	0x0	EMRS_DEV_SELECTN: [PMC] Active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
27	0x0	USE_EMRS_EXT_CNT: [PMC] 0 = USE_MRS_LONG_CNT will select between SHORT(0)/LONG(1), 1 = USE_MRS_LONG_CNT will select between EXT1(0)/EXT2(1).
26	SHORT	USE_EMRS_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
21:20	0x0	EMRS_BA: [PMC] Set to 0x1 for EMRS (and where applicable, 0x2 for EMRS2, and 0x3 for EMRS3).
13:0	0x0	EMRS_ADR: [PMC] Mode-register data to be written.

16.7.2.49 EMC_REF_0

Command Trigger: Refresh

Note: The REF register allows software 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: 0xd4 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxx00000000xxxxxx00)

Bit	Reset	Description
31:30	0x0	REF_DEV_SELECTN: active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
15:8	0x0	REF_NUM: perform (REF_NUM + 1) refresh cycles.
1	0x0	REF_NORMAL: 0 = execute first refresh immediately (this is use during DRAM initialization). 1 = execute refresh through normal refresh mechanism (this should be used in normal operation).
0	0x0	REF_CMD: causes the hardware to perform a REFRESH to all DRAM banks.

16.7.2.50 EMC_PRE_0

Command Trigger: Precharge-All

The PRE register allows software to issue a precharge all command. This command may be used to ensure proper DRAM initialization.

Boot requirements:

- This register triggers a precharge-all command. PRE_CMD should be written with 0x1 during cold boot to trigger refresh commands during DRAM initialization.
- If per-device DPD is used, the PRE_DEV_SELECTN field should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0xd8 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
31:30	0x0	PRE_DEV_SELECTN: active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
0	0x0	PRE_CMD: causes the hardware to perform a PRECHARGE to all DRAM banks.

16.7.2.51 EMC_NOP_0

Command Trigger: NOP

The NOP register allows software to issue an explicit NOP command. This command may be used to ensure proper DRAM initialization.

Boot requirements:

- This register triggers a no-operation command. NOP_CMD should be written with 0x1 during cold boot to trigger no-op commands during DRAM initialization.

Offset: 0xdc | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	Reset	Description
31:30	0x0	NOP_DEV_SELECTN: active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
0	0x0	NOP_CMD: causes the hardware to perform a NOP to all DRAM banks.

16.7.2.52 EMC_SELF_REF_0

Command Trigger: SELF REFRESH

The SELF_REF register allows software to issue self-refresh commands.

Do not program this register when a clock change sequence is on-going. Check the CLKCHANGE_COMPLETE_INT value if not sure.

Offset: 0xe0 | Read/Write: R/W | Reset: 0x00000000 (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

16.7.2.53 EMC_DPD_0

Command Trigger: Deep Power Down

The DPD register allows software to issue a deep power down command.

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

Bit	Reset	Description
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

16.7.2.54 EMC_MRW_0

Command Trigger: MRW

Mode Register Write: LPDDR3-only version of MRS/EMRS

Boot requirements:

- This register triggers a mode register write command. Multiple mode-register writes may be required by the cold boot sequence. Such commands should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0xe8 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xx00000000xxxxxxxx00000000

Bit	Reset	Description
31:30	0x0	MRW_DEV_SELECTN: [PMC] Active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW_EXT_CNT: [PMC] 0 = USE_MRS_LONG_CNT will select between SHORT(0)/LONG(1), 1 = USE_MRS_LONG_CNT will select between EXT1(0)/EXT2(1).
26	SHORT	USE_MRW_LONG_CNT: [PMC] Indicate to use long or short MRS wait count.
23:16	0x0	MRW_MA: [PMC] Register address
7:0	0x0	MRW_OP: [PMC] Data to be written

16.7.2.55 EMC_MRR_0

Command Trigger: MRR

Mode Register Read: LPDDR3 only

Sequence

1. Read MRR until EMC_STATUS.MRR_DIVLD=0 (ok to skip if it is sure there are no pending MRR reads)

2. Write this register with the desired addr (MA) and device (DEV_SELECTN), device needs to be either DEV0 or DEV1: writing to both is illegal

3. Poll EMC_STATUS.MRR_DIVLD=1, or if using interrupt, wait for MRR_DIVLD_INT

4. Read back MRR. The value is in MRR_DATA field.

Note: It is okay to issue new MRR requests while there are on-going requests. For example, to issue 3 MRR requests, follow these steps: 1,2,2,2,3,4,3,4,3,4. Data read back in step 4 is in the same order requested in step 2.

To make sure the EMC is available for new MRR requests, poll for EMC_STATUS.MRR_FIFO_SPACE > 0 before step 2.

If using 2 x16 DRAM, MRR_DATA[15:8] can be used to store MRR from the second DRAM on the same CS (configured via CFG_3.MRR_BYTESEL_X16)

Offset: 0xec | Read/Write: R/W | Reset: 0x0000XXXX (0b00xx00xx00000000xxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31:30	RW	0x0	MRR_DEV_SELECTN: active-low chip-select, choose which device to send the command to. (enum for safety). 0 = ILLEGAL 1 = DEV1 2 = DEV0 3 = RESERVED
27	RW	0x0	USE_MRR_EXT_CNT: 0 = USE_MRS_LONG_CNT will select between SHORT(0)/LONG(1), 1 = USE_MRS_LONG_CNT will select between EXT1(0)/EXT2(1).
26	RW	SHORT	USE_MRR_LONG_CNT: Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
23:16	RW	0x0	MRR_MA: register address
15:0	RO	X	MRR_DATA: data returned

16.7.2.56 EMC_CMDQ_0

Command Queue Depth Register

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This arbitration configuration register should be parameterized in the BCT and written by the OS during cold boot and warm boot.

Offset: 0xf0 | Read/Write: R/W | Reset: 0x10004408 (0bxxx10000xxxxxxxxx100x100xxx01000)

Bit	Reset	Description
28:24	0x10	RW_WD_DEPTH
14:12	0x4	PRE_DEPTH
10:8	0x4	ACT_DEPTH
4:0	0x8	RW_DEPTH

16.7.2.57 EMC_MC2EMCQ_0

Command Queue Depth Register

Boot requirements:

- This arbitration configuration register should be parameterized in the BCT and written by the OS during cold boot and warm boot.

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0xf4 | Read/Write: R/W | Reset: 0x06000404 (0bxxxx0110xxxxxxxxxxxxx100xxxxx100)

Bit	Reset	Description
27:24	0x6	MCWD_DEPTH
10:8	0x4	MCACT_DEPTH
2:0	0x4	MCREQ_DEPTH

16.7.2.58 EMC_XM2DQSPADCTRL3_0

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0xf8 | Read/Write: R/W | Reset: 0x20820800 (0bx01000x01000x01000x01000xx0xxxx0)

Bit	Reset	Description
30:26	0x8	EMC2PMACRO_CFG_XM2DQS_BYTE3_VREF_DQS: [PMC]
24:20	0x8	EMC2PMACRO_CFG_XM2DQS_BYTE2_VREF_DQS: [PMC]
18:14	0x8	EMC2PMACRO_CFG_XM2DQS_BYTE1_VREF_DQS: [PMC]
12:8	0x8	EMC2PMACRO_CFG_XM2DQS_BYTE0_VREF_DQS: [PMC]
5	0x0	EMC2PMACRO_CFG_XM2DQS_E_VREF_DQS: [PMC]
0	0x0	EMC2PMACRO_CFG_XM2DQS_E_STRPULL_DQS: [PMC]

16.7.2.59 EMC_FBIO_SPARE_0

FBIO Spare Register

Note: CFG_FBIO_SPARE_0[1] = FBIO_SPARE[1] = lock further change to ADDR_SWIZZLE_STACK* and ADR_CFG. registers.

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- CFG_FBIO_SPARE_3 should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x100 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	CFG_FBIO_SPARE_3
23:16	0x0	CFG_FBIO_SPARE_2
15:8	0x0	CFG_FBIO_SPARE_1
7:0	0x0	CFG_FBIO_SPARE_0

16.7.2.60 EMC_FBIO_CFG5_0

FBIO Configuration Register

The trimmer value may be overridden by setting using MULT==0 and OFFS = value << 4.

The FBIO_CFG5 register controls the FBIO I/O cells.

The following fields are shadowed: DIFFERENTIAL_DQS, CTT_TERMINATION, DQS_PULLD, CMD_2T_TIMING.

Writes to these fields will not take effect until the active value is updated via TIMING_UPDATE or (if enabled) CLKCHANGE_REQ.

Boot requirements:

- This register (except for field DISABLE_CONCURRENT_AUTOPRE) should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register (except for field DISABLE_CONCURRENT_AUTOPRE) should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x104 | Read/Write: R/W | Reset: 0x10400815 (0bxxx1x00001000000000100000010101) | Default: 0x00400010

Bit	Reset	SW Default	Description
28	ENABLED	NONE	MASK_PUTERM_N_DQS_PULLD_DURING_ZQCAL: [PMC] When this bit is enabled; it deasserts DQ/DQS puterm and DQS pulld during ZQ calibration in hardware. 0 = DISABLED 1 = ENABLED
26	DISABLED	NONE	CMD_BUS_RETURN_TO_ONE: [PMC] this bit drives the cmd bus back to a one as a resting stage (legacy Tegra device behavior). Power is saved if this is not done (Tegra K1 behavior). Shadowed 0 = DISABLED 1 = ENABLED
25	DISABLED	NONE	LPDDR3_DRAM: [PMC] Enables differentiation between lpddr3 DRAM protocol. Should only be set if LPDDR3 is selected with DRAM_TYPE. 0 = DISABLED 1 = ENABLED
24	DISABLED	NONE	LPDDR3_WR_PREAMBLE_TOGGLE: [PMC] enable LPDDR3 write preamble toggle. 0 = DISABLED 1 = ENABLED
23:20	0x4	0x4	ERR_RD_BUBBLE: [PMC] Number of bubbles to be inserted in CMDQ after every error-Read. A dummy read is placed in CMDQ for each error read. This field controls the number of emcclk bubbles inserted for each dummy read. It is expected that this should be set to BL/2. Without this, ret_stat_fifo will build up and block non-error reads. This could cause ret_data_fifo to overflow as it is only 8 deep (area savings) where as ret_stat_fifo is 20 deep. Shadowed
19:16	DDR3L_4X16	NONE	CMD_MAPPING:[PMC_SECURE] Shadowed. The command mapping is locked from any additional updates once fbio_spare[1] is set. The MID_DDR3L_4X16_96_2T2B_SWONLY is a placeholder for a software-based cmd mapping based on MID_DDR3L_4X16_96_4TOP. The hardware register should always be programmed to MID_DDR3L_4X16_96_4TOP. 0 = DDR3L_4X16 1 = DSC_LPDDR3_1X64_253 2 = DSC_LPDDR2_2X32_134 3 = DSC_LPDDR3_2X32_178 4 = DSC_DDR3L_2X32_136 5 = DSC_DDR3L_4X16_96_2X2H 6 = DSC_DDR3L_4X16_96_2X2V 7 = DSC_DDR3L_4X16_96_2TBV 8 = DSC_DDR3L_4X16_96_2TBH 9 = DSC_DDR3L_4X16_96_1X4_2LYR 10 = MID_DDR3L_4X16_96_4TOP 11 = MID_DDR3L_2X32_136 12 = MID_DDR3L_8X8_78_8TOP 13 = MID_DDR3L_4X16_96_4BOT 14 = MID_DDR3L_4X16_96_2T2B_SWONLY
15:13	NORMAL	NONE	EMC2PMACRO_CFG_QUSE_MODE: [PMC] Select QUSE mode. Shadowed. (NORMAL = off-chip) (ALWAYS_ON = requires DQS PULL) (INTERNAL_LPBK selects on-chip loopback) (DIRECT_QUSE = directly controlled QUSE). 0 = NORMAL 1 = ALWAYS_ON 2 = INTERNAL_LPBK 3 = PULSE_INT

Bit	Reset	SW Default	Description
			4 = RESERVED 5 = DIRECT_QUSE 6 = RESERVED1
12	DISABLED	NONE	CMD_2T_TIMING: [PMC] Enable 2T command timing (RAS/CAS/WE/ROW/A/BA). Shadowed. 0 = DISABLED 1 = ENABLED
11	ENABLED	NONE	CFG_QUSE_EXTEND_HALF_CLK: [PMC] Extend the QUSE window by half clk in DIRECT_QUSE mode. Shadowed. 0 = DISABLED 1 = ENABLED
10	DISABLED	NONE	DISABLE_CONCURRENT_AUTOPRE: [PMC] Disables reads/writes to a device until the precharge command has been issued by the DRAM internally. 0 = DISABLED 1 = ENABLED
9	DISABLED	NONE	DQS_PULLD: [PMC] Enables pull-downs on DQS lines (and pull-ups on DQS_N if DIFFERENTIAL_DQS). Shadowed. 0 = DISABLED 1 = ENABLED
8	DISABLED	NONE	CTT_TERMINATION: [PMC] Enables CTT_TERMINATION mode in pads (DDR2 support). Shadowed. 0 = DISABLED 1 = ENABLED
7	DISABLED	NONE	DIFFERENTIAL_DQS: [PMC] Enables differential signaling on DQS strobes. Shadowed. 0 = DISABLED 1 = ENABLED
6	DISABLED	NONE	CFG_E_PUTERM_DQ: [PMC] Enables E_PUTERM_DQ mode in pads (DDR3 support). In the IOBRICK, the E_PUTERM_DQ mode overrides CTT_TERMINATION mode. Shadowed. 0 = DISABLED 1 = ENABLED
5	LOWER	LOWER	DRAM_DATA_SWZL: [PMC] Specifies which half of the data bytes are used in X32 mode. LOWER:B0-B3 UPPER:B4-B7 0 = LOWER 1 = UPPER
4	X64	X64	DRAM_WIDTH: [PMC] Specifies DRAM width. Platform specific. 0 = X32 1 = X64
3:2	BURST8	NONE	DRAM_BURST: [PMC] Specifies the burst length to use for the attached device(s). ON_THE_FLY BC4/BL8 is only use for DDR3. Shadowed. 0 = BURST4 1 = BURST8 2 = ON_THE_FLY 3 = RESERVED
1:0	DDR1	NONE	DRAM_TYPE: [PMC] Specifies which DRAM protocol to use for the attached device(s). 0 = DDR3 1 = DDR1 2 = LPDDR3 3 = DDR2

16.7.2.61 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.

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x114 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx010)

Bit	Reset	Description
2:0	0x2	CFG_QUSE_LATE: [PMC]

16.7.2.62 EMC_CFG_RSV_0

EMC Configuration Reserved

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x120 | Read/Write: R/W | Reset: 0xff00ff00 (0b11111111000000001111111100000000)

Bit	Reset	Description
31:24	0xff	CFG_RESERVED_BYTE3
23:16	0x0	CFG_RESERVED_BYTE2
15:8	0xff	CFG_RESERVED_BYTE1
7:0	0x0	CFG_RESERVED_BYTE0

16.7.2.63 EMC_ACPD_CONTROL_0

Threshold for ACPD

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x124 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	ACPD_THRESHOLD: [PMC] number of idle cycles to wait before allowing power-down entry

16.7.2.64 EMC_EMRS2_0

Command Trigger: EMRS2

The EMRS2 register allows software to issue an EMRS2 command.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0x12c | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xxxx00xxxxxx00000000000000)

Bit	Reset	Description
31:30	0x0	EMRS2_DEV_SELECTN: [PMC] active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
27	0x0	USE_EMRS2_EXT_CNT: [PMC] 0=>USE_MRS_LONG_CNT will select between SHORT(0)/LONG(1), 1=>USE_MRS_LONG_CNT will select between EXT1(0)/EXT2(1).
26	SHORT	USE_EMRS2_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
21:20	0x0	EMRS2_BA: [PMC] Set to 0x1 for EMRS (and where applicable, 0x2 for EMRS2, and 0x3 for EMRS3).
13:0	0x0	EMRS2_ADR: [PMC] mode-register data to be written.

16.7.2.65 EMC_EMRS3_0

Command Trigger: EMRS2

The EMRS3 register allows software to issue an EMRS3 command.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0x130 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xxxx00xxxxxx00000000000000)

Bit	Reset	Description
31:30	0x0	EMRS3_DEV_SELECTN: [PMC] active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
27	0x0	USE_EMRS3_EXT_CNT: [PMC] 0=>USE_MRS_LONG_CNT will select between SHORT(0)/LONG(1), 1=>USE_MRS_LONG_CNT will select between EXT1(0)/EXT2(1).
26	SHORT	USE_EMRS3_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
21:20	0x0	EMRS3_BA: [PMC] Set to 0x1 for EMRS (and where applicable, 0x2 for EMRS2, and 0x3 for EMRS3).
13:0	0x0	EMRS3_ADR: [PMC] mode-register data to be written.

16.7.2.66 EMC_MRW2_0

Command Trigger: MRW2

Mode Register Write: LPDDR3 -only version of MRS/EMRS

Boot requirements:

This register triggers a mode register write command. Multiple mode-register writes may be required by the cold boot sequence. Such commands should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0x134 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xx00000000xxxxxxxx00000000)

Bit	Reset	Description
31:30	0x0	MRW2_DEV_SELECTN: [PMC] active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW2_EXT_CNT: [PMC] 0=>USE_MRS_LONG_CNT will select between SHORT(0)/LONG(1), 1=>USE_MRS_LONG_CNT will select between EXT1(0)/EXT2(1).
26	SHORT	USE_MRW2_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
23:16	0x0	MRW2_MA: [PMC] register address
7:0	0x0	MRW2_OP: [PMC] data to be written

16.7.2.67 EMC_MRW3_0

Command Trigger: MRW3

Mode Register Write: LPDDR3 -only version of MRS/EMRS

Boot requirements:

This register triggers a mode register write command. Multiple mode-register writes may be required by the cold boot sequence. Such commands should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0x138 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xx00000000xxxxxxxx00000000)

Bit	Reset	Description
31:30	0x0	MRW3_DEV_SELECTN: [PMC] active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW3_EXT_CNT: [PMC] 0=>USE_MRS_LONG_CNT will select between SHORT(0)/LONG(1), 1=>USE_MRS_LONG_CNT will select between EXT1(0)/EXT2(1).
26	SHORT	USE_MRW3_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
23:16	0x0	MRW3_MA: [PMC] register address
7:0	0x0	MRW3_OP: [PMC] data to be written

16.7.2.68 EMC_MRW4_0

Command Trigger: MRW4

Mode Register Write: LPDDR3 -only version of MRS/EMRS

Boot requirements:

This register triggers a mode register write command. Multiple mode-register writes may be required by the cold boot sequence. Such commands should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0x13c | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xx00000000xxxxxxxx00000000)

Bit	Reset	Description
31:30	0x0	MRW4_DEV_SELECTN: [PMC] active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW4_EXT_CNT: [PMC] 0=>USE_MRS_LONG_CNT will select between SHORT(0)/LONG(1), 1=>USE_MRS_LONG_CNT will select between EXT1(0)/EXT2(1).
26	SHORT	USE_MRW4_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT

Bit	Reset	Description
		1 = LONG
23:16	0x0	MRW4_MA: [PMC] register address
7:0	0x0	MRW4_OP: [PMC] data to be written

16.7.2.69 EMC_CLKEN_OVERRIDE_0

Second-Level Clock Enable Override Register

Offset: 0x140 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0xx000x)

Bit	Reset	Description
31	DISABLED	OBS_BUS_CLKEN: 0 = DISABLED 1 = ENABLED
6	CLK_GATED	STATS_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
3	CLK_GATED	RR_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
2	CLK_GATED	DRAMC_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
1	CLK_GATED	CMDQ_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED

16.7.2.70 EMC_R2R_0

DRAM Timing Parameter

This register is shadowed: see usage note at the top of Section 16.7.2

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: 0x144 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	R2R:

16.7.2.71 EMC_W2W_0

DRAM Timing Parameter

This register is shadowed: see usage note at the top of Section 16.7.2

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: 0x148 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	W2W:

16.7.2.72 EMC_EINPUT_0

DRAM Timing Parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x14c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5:0	0x0	EINPUT: [PMC] specifies when to assert EINPUT for a read, should normally be the same as QUSE.

16.7.2.73 EMC_EINPUT_DURATION_0

DRAM Timing Parameter

The minimum for EINPUT_DURATION is 4.

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x150 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00100)

Bit	Reset	Description
4:0	0x4	EINPUT_DURATION: [PMC] Specifies how long the EINPUT should be asserted.

16.7.2.74 EMC_PUTERM_EXTRA_0

DRAM Timing Parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x154 | Read/Write: R/W | Reset: 0x00010000 (0bxxxxxxxxxx000001xxxxxxxxxxxxxxxx)

Bit	Reset	Description
21:16	0x1	PUTERM: [PMC] tells the chip when to assert dynamic PUTERM for read return data.

16.7.2.75 EMC_TCKESR_0

DRAM Timing Parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x158 | Read/Write: R/W | Reset: 0x0000003e (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111110)

Bit	Reset	Description
5:0	0x3e	TCKESR: [PMC] Specify minimum low CKE pulse width for self-refresh mode.

16.7.2.76 EMC_TPD_0

DRAM Timing Parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x15c | Read/Write: R/W | Reset: 0x0000003e (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111110)

Bit	Reset	Description
5:0	0x3e	TPD: [PMC] specify minimum low CKE pulse width for power-down mode.

16.7.2.77 EMC_AUTO_CAL_CONFIG_0

Auto-Calibration Settings for EMC Pads

Boot requirements:

- This register (except for AUTOCAL_SLW_OVERRIDE and AUTO_CAL_START(trigger)) should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register (except for AUTOCAL_SLW_OVERRIDE and AUTO_CAL_START(trigger)) should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2a4 | Read/Write: R/W | Reset: 0x00f10000 (0b0000xx001111x001xxx00000xxx00000)

Bit	R/W	Reset	Description
31	RO	0x0	AUTO_CAL_START: [PMC] Writing a one to this bit starts the calibration state machine. This bit must be set even if the override is set in order to latch in the override value.
30	RW	0x0	AUTO_CAL_OVERRIDE: [PMC] 0 (normal operation): use AUTO_CAL_PU/PD_OFFSET as an offset to the calibration state machine setting 1 (override) : use AUTO_CAL_PU/PD_OFFSET register values directly
29	RW	DISABLED	AUTO_CAL_ENABLE: [PMC] 1 (normal operation): use EMC generated pullup/down (override or autocal) 0 (disabled): use emc2tmc_cfg* register settings for pullup/down 0 = DISABLED 1 = ENABLED
28	RW	0x0	AUTOCAL_SLW_OVERRIDE: [PMC] 0 (Normal operation) pad DRVDN/UP_SLWR/F tied to AUTO_CAL output DRDVDN/UP_SLWR/F[3:0] = AUTO_CAL_PULLDOWN/UP[4:1] 1 (override) use EMC2TMC_CFG*_DRVDN/UP_SLWR/F pins to control pad slew inputs
25:20	RW	0xf	AUTO_CAL_E_CAL_UPDATE: [PMC] wait time in EMC clocks between clk cal code change, and E_CAL_UPDATE assertion is AUTO_CAL_E_CAL_UPDATE - 4. Maximum value is 31.
18:16	RW	0x1	AUTO_CAL_STEP: [PMC] calibration step interval (in microseconds)
12:8	RW	0x0	AUTO_CAL_PD_OFFSET: [PMC] 2's complement offset for the pull-down value
4:0	RW	0x0	AUTO_CAL_PU_OFFSET: [PMC] 2's complement offset for the DQ/DQS pull-up value

16.7.2.78 EMC_AUTO_CAL_INTERVAL_0

EMC Pad Calibration Interval

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
20:0	0x0	AUTO_CAL_INTERVAL: [PMC] 0: Calibrate once. Otherwise, auto-calibration occurs at intervals equivalent to the programmed number of microseconds.

16.7.2.79 EMC_AUTO_CAL_STATUS_0

EMC Pad Calibration Status

Offset: 0x2ac | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	AUTO_CAL_ACTIVE: One when auto calibrate is active - valid only after auto calibrate sequence has completed (EMC_CAL_ACTIVE == 0)
28:24	X	AUTO_CAL_PULLDOWN_ADJ: Pull-down code sent to pads
20:16	X	AUTO_CAL_PULLUP_ADJ: Pull-up code sent to pads
12:8	X	AUTO_CAL_PULLDOWN: Pull-down code generated by auto-calibration
4:0	X	AUTO_CAL_PULLUP: Pull-up code generated by auto-calibration

16.7.2.80 EMC_REQ_CTRL_0

Request Status/Control

When either STALL_ALL_READS and/or STALL_ALL_WRITES is asserted, the stalling of read and/or write requests will not take effect until the status bit from EMC_STATUS register's NO_OUTSTANDING_TRANSACTIONS field is asserted.

Offset: 0x2b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	STALL_ALL_WRITES: Stall incoming write transactions
0	0x0	STALL_ALL_READS: Stall incoming read transactions

16.7.2.81 EMC_EMC_STATUS_0

EMC State-Machine Status

DRAM_IN_POWERDOWN, DRAM_IN_SELF_REFRESH, DRAM_IN_DPD: active high signal indicating current DRAM status for each of these modes, with 1 status bit per device, bit[0] = dev0 status, bit[1] = dev1 status.

Example: DRAM_IN_SELF_REFRESH = 0x3, both devices are in self-refresh, DRAM_IN_DPD= 0x2 would indicate only dev1 is in deep-power-down mode.

> **Note:** If EMC is reset or powered down, the actual DRAM state could be different than indicated by these status bits. These bits do not reflect manually entered/exited powerdown or self-refresh (via use of PIN_CKE).

Offset: 0x2b4 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
27:26	X	ACPD_FSM_IDLE: Indicate dev[n] power-down FSM is idle.
25:24	X	DSR_FSM_IDLE: Indicate dev[n] dynamic self refresh FSM is idle.
23	X	TIMING_UPDATE_STALLED: Indicates timing update triggered by TIMING_CONTROL.TIMING_UPDATE is not yet completed.
22	X	ZQ_FSM_IDLE: Indicates auto ZQ state machine is idle.
21	X	CFG_ZQ_ACTIVE: Indicates ZQ configuration access is active.
20	X	MRR_DIVLD: MRR data available for reading
19:16	X	MRR_FIFO_SPACE: MRR FIFO space available

Bit	Reset	Description
13:12	X	DRAM_IN_DPD: dev[n] has been put into deep powerdown state
9:8	X	DRAM_IN_SELF_REFRESH: dev[n] has been put into self-refresh (will remain until SR exit cmd).
5:4	X	DRAM_IN_POWERDOWN: dev[n] has entered powerdown state (incoming req's will awaken if not stalled)
2	X	NO_OUTSTANDING_TRANSACTIONS: All non-stalled requests have completed
0	X	EMC_REQ_FIFO_EMPTY: Request FIFO is empty

16.7.2.82 EMC_CFG_2_0

EMC Configuration

Clock change sequencing: Once the divider is reprogrammed, CAR signals to EMC that a clock change is pending. If enabled, EMC stalls incoming requests, drains outstanding requests, and, if CLKCHANGE_(PD|SR)_ENABLE is enabled, puts DRAM into power-down (or self-refresh) before signalling to CAR that it is idle and ready for the change to happen. CAR will then change the divider/pll reprogramming. Once complete, EMC updates its shadow registers (assuming they may have been reprogrammed for new clock setting), unstalls requests, and resumes operation with new clock settings.

Some fields of this register are shadowed: see usage note at the top of Section 16.7.2

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.

Offset: 0x2b8 | Read/Write: R/W | Reset: 0x00008057 (0b0000000000xxxxxx1000000001010111)

Bit	Reset	Description
31	DISABLED	DRAMC_PRE_B4_ACT: [PMC] Diagnostic bit, gives priority to activates over precharges, determining which (precharge/activate) is processed first if both are pending and unblocked. 0 = DISABLED 1 = ENABLED
13	DISABLED	USE_PER_DEVICE_DLY_TRIM_OB: [PMC] Diagnostic bit. Shadowed. Allows using different outbound delay trim values per device. When enabled, WDV must be programmed to >= 2 and WEXT >=5. 0 = DISABLED 1 = ENABLED
12	DISABLED	USE_PER_DEVICE_DLY_TRIM_IB: [PMC] Diagnostic bit. Shadowed. Allows using different inbound delay trim values per device. When enabled, REXT/RDV/QUSE must be programmed to >= 2. 0 = DISABLED 1 = ENABLED
11	0x0	DIS_CNTR_WITH_CFG_TIMING_UPDATE: [PMC] Diagnostic bit. 1 = Disable reset of timing parameter counters with TIMING_CONTROL.TIMING_UPDATE.
9:8	0x0	Reserved
7	0x0	EARLY_TRFC_8_CLK: [PMC] Diagnostic bit. Shadowed. 1 = 8 clocks early 0 = 16 clocks early.
6	0x1	DIS_STP_OB_CLK_DURING_NON_WR: [PMC] 1 = Disables stopping outbound data clock to I/O during non-write transactions.
5:3	0x2	ZQ_EXTRA_DELAY: [PMC] Additional delay to push out ZQCMD based on tODToff (maximum) & frequency.
2	0x1	REF_AFTER_SREF: [PMC] 1 = Enables trigger of refresh after exiting self-refresh.

Bit	Reset	Description
1	ENABLED	CLKCHANGE_PD_ENABLE: [PMC] Forces DRAM into power-down during CLKCHANGE. 0 = DISABLED 1 = ENABLED
0	ENABLED	CLKCHANGE_REQ_ENABLE: [PMC] Allows the EMC and CAR to handshake on PLL divider/source changes. 0 = DISABLED 1 = ENABLED

16.7.2.83 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 the top of Section 16.7.2

Boot requirements:

- This register (except for field CFG_DLL_ALARM_DISABLE and DLL_RESET (trigger) and CFG_DLL_USE_OVERRIDE_UNTIL_LOCK(trigger)) should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register (except for field CFG_DLL_ALARM_DISABLE and DLL_RESET (trigger) and CFG_DLL_USE_OVERRIDE_UNTIL_LOCK(trigger)) should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2bc | Read/Write: R/W | Reset: 0xXX0000Xd (0bx0000x0000000000000000011x11x1)

Bit	R/W	Reset	Description
31	RO	X	CFG_DLL_USE_OVERRIDE_UNTIL_LOCK: [PMC] Writing 1 to this register causes override_val to be used in place of DLL output until DLL_LOCK/DLL_LOCK_TIMEOUT is obtained. Takes effect on next shadow update.
30	RO	0x0	DLL_RESET: [PMC] Writing 1 to this register will send reset pulse to DLL's on next shadow update. Must reset DLLs when changing clock frequency by factor >= 2. Reset occurs at next shadow update.
29:28	RW	0x0	CFG_DLL_LOCK_LIMIT: [PMC] Diagnostic in case DLL has problems locking. Shadowed. DLL will be treated as locked after LIMIT µs. Counter is reset with DLL_RESET (from above) or with each periodic update (if using RUN_PERIODIC). Settings are: 00: LIMIT = 16 µs 01: LIMIT = 64 µs 10: LIMIT = 128 µs 11: LIMIT = 512 µs
27	RW	0x0	CFG_DLL_ALARM_DISABLE: [PMC] Diagnostic bit. Shadowed. Disable override of DLL logic when DLL_ALM is set (otherwise overrides DLI to 0x3FF).
26	RO	X	CFG_DLL_UPDATE_AT_NXT_REFRESH: Writing 1 to this register causes the DLCELL update to occur at the next REFRESH interval. Ordinarily, updates are only performed if the DLL value is updated. This mechanism allows updates to the XFORM_MULT/OFF to be taken at the next REFRESH.
25:16	RW	0x0	CFG_DLL_OVERRIDE_VAL: [PMC] Value to use in place of DLI output if CFG_DLL_OVERRIDE_EN is set or prior to lock in RUN_TIL_LOCK mode. Shadowed.
15	RW	0x0	CFG_DLL_TESTEN: [PMC] Enable DLL test mode
14	RW	0x0	CFG_DLL_QUSE_TESTOUT: [PMC] Enable DLL TESTOUT on QUSE pads
13:12	RW	0x0	CFG_DLL_TESTSEL: [PMC] Select DLL test output

Bit	R/W	Reset	Description
11:8	RW	0x0	CFG_DLL_UDSET: [PMC] DLL Loop filter control (2^(udset+3)). Shadowed.
7:6	RW	RUN_TIL_LOCK	CFG_DLL_MODE: [PMC] Controls how frequently DLL runs. Shadowed. 0 = RUN_CONTINUOUS: DLL will run continuously. This option will consume the most power. Once LOCK/TIMEOUT occurs, trimmers will be updated at REFRESH or shadow update. 1 = RUN_TIL_LOCK: after DLL_RESET is set, DLL will run until it has locked/timed out, then be disabled. Trimmers updated at following REFRESH/shadow update. 2 = RUN_PERIODIC: after DLL_LOCK/TIMEOUT, DLL will be disabled and re-enabled after CFG_DLL_RUN_PERIOD µs to track delay changes (temperature/voltage). 3 = RESERVED
5	RW	0x1	CFG_DLL_LOWSPEED: [PMC] Enable DLL for use with low-speed EMCCLK operation (< 200 MHz). Shadowed. The DLL does power optimization by itself. This control should stay at 1.
4	RO	X	CFG_DLL_STALL_RW_UNTIL_LOCK: [PMC] Writing 1 to this register will cause the EMC to stall all RW traffic until the DLL locks. Takes effect on the next shadow update.
3	RW	0x1	CFG_DLL_STALL_ALL_TRAFFIC: [PMC] Enables stalling of all DRAM traffic while undergoing dll_stall condition;otherwise only block reads/writes are stalled during dll_stall. Shadowed.
2	RW	ENABLED	CFG_DLL_OVERRIDE_EN: [PMC] Override DLL's DLI output with OVERRIDE_VAL (still uses mult/offset). Shadowed. 0 = DISABLED 1 = ENABLED
0	RW	ENABLED	CFG_DLL_EN: [PMC] Enables digital DLL. Shadowed. 0 = DISABLED 1 = ENABLED

16.7.2.84 EMC_CFG_DIG_DLL_PERIOD_0

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x2c0 | Read/Write: R/W | Reset: 0x00008000 (0bxxxxxxxxxxxxxxxx1000000000000000)

Bit	Reset	Description
15:0	0x8000	CFG_DLL_RUN_PERIOD: [PMC] If CFG_DLL_MODE == RUN_PERIODIC, this specifies the interval between runs in microseconds.

16.7.2.85 EMC_DIG_DLL_STATUS_0

Digital DLL Status

Digital DLL Status bits directly from the DLL

Offset: 0x2c8 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15	X	DLL_LOCK
14	X	DLL_ALARM
13	X	DLL_LOCK_TIMEOUT
9:0	X	DLL_OUT

16.7.2.86 EMC_RDV_MASK_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2cc | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	RDV_MASK: [PMC] This should be programmed to RDV.

16.7.2.87 EMC_WDV_MASK_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2d0 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111)

Bit	Reset	Description
3:0	0xf	WDV_MASK: [PMC] This should be programmed to WDV.

16.7.2.88 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.

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x2d8 | Read/Write: R/W | Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0011)

Bit	Reset	Description
3:0	0x3	CTT_DURATION: [PMC] Determines how long CTT remains enabled during reads. CTT determines when it will be enabled. DDR3: CTT_TERMINATION == 1 ? 4 : 0 LPDDR3: 0

16.7.2.89 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

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register (except for fields TERM_DRVDN and TERM_DRVUP) should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register (except for fields TERM_DRVDN and TERM_DRVUP) should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2dc | Read/Write: R/W | Reset: 0xXX0XX802 (0b0xxxxxxxxxxxxxxxxxx01000xxxxx010)

Bit	R/W	Reset	Description
31	RW	DISABLED	TERM_OVERRIDE: [PMC] 0 = DISABLED 1 = ENABLED
28:24	RO	X	TERM_DRVUP
19:15	RO	X	TERM_DRVDN
12:8	RW	0x8	TERM_OFFSET: [PMC]
2:0	RW	0x2	TERM_SLOPE: [PMC]

ZQ Calibration Registers -- LPDDR3/DDR3

1. Periodic mode

Allows a ZQ calibration command to be sent periodically to DRAM. After ZCAL_INTERVAL, ZCAL_MRW_CMD will be sent to each DRAM, 1 at a time, followed by ZCAL_WAIT_CNT interval in which no other commands will be allowed to be sent to either DRAM. So if ZQ_MRW_DEV_SELECTN == 2'b00, it would send the MRW command to dev0, wait ZCAL_WAIT_CNT, send the command to dev1, wait ZCAL_WAIT_CNT, resume normal operation. It will then wait ZCAL_INTERVAL microseconds before repeating the procedure. The ZQ calibration command will not be sent to devices that are 1) unpopulated and 2) either in or about to enter self-refresh or DPD.

To enable periodic ZQ calibration, program ZCAL_INTERVAL and ZCAL_WAIT_CNT to be non-zero, as well as ZCAL_MRW_CMD to contain the command and device selection, followed by TIMING_CONTROL to latch those three registers.

Note that ZCAL_WAIT_CNT will be used instead of MRS_WAIT_CNT for delaying the subsequent DRAM commands after the ZQ calibration commands, even though in LPDDR3 a ZQ calibration command is also an MRW command.

Disable this feature by setting ZCAL_INTERVAL.ZCAL_REF_INTERVAL = 0, followed by TIMING_CONTROL to latch it.

2. One-shot mode

Allows ZQ calibration command to be sent to DRAM.

To send a one-shot command, keep ZCAL_INTERVAL and ZCAL_WAIT_CNT equal to zero, program ZCAL_MRW_CMD with only one device selected, then program TIMING_CONTROL to latch in those registers, and finally program

ZCAL_ONE_SHOT to 1 to trigger the one-shot command. If more than one device needs to be calibrated, wait the ZQ calibration time and repeat the steps above with the other device selection bit enabled in ZCAL_MRW_CMD.

Do not send a one-shot ZQ calibration command when periodic mode is enabled.

16.7.2.90 EMC_ZCAL_INTERVAL_0

Configure ZQ Calibration

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:10	0x0	ZCAL_INTERVAL_HI: [PMC] Combined with ZCAL_INTERVAL_LO specifies the number of microseconds to wait between issuance of ZCAL_MRW_CMD. If 0, ZCAL is disabled and internal counter will be reset.
9:0	0x0	ZCAL_INTERVAL_LO [PMC]

16.7.2.91 EMC_ZCAL_WAIT_CNT_0

Configure ZQ Calibration

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:0	0x0	ZCAL_WAIT_CNT: [PMC] Number of EMC clocks to wait before issuing any commands after sending ZCAL_MRW_CMD.

16.7.2.92 EMC_ZCAL_MRW_CMD_0

Configure ZQ Calibration

This register is shadowed: see usage note at the top of Section 16.7.2

(However, it does not obey READ_MUX/WRITE_MUX non-default values.)

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2e8 | Read/Write: R/W | Reset: 0xc0000000 (0b11xxxxxx00000000xxxxxxxx00000000)

Bit	Reset	Description
31:30	0x3	ZQ_MRW_DEV_SELECTN: [PMC] Active-low chip-select, 0x0 applies command to both

Bit	Reset	Description
		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

16.7.2.93 EMC_ZQ_CAL_0

Trigger a Single ZQ Calibration

This register issues a ZQ calibration command for DDR3.

Offset: 0x2ec | Read/Write: R/W | Reset: 0x00000010 (0b00xxxxxxxxxxxxxxxxxxxxxxxxx1xxx0)

Bit	Reset	Description
31:30	0x0	ZQ_CAL_DEV_SELECTN: Active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1, 0x3 for neither device (0x0, for both devices, is not allowed if the ZQ resister is shared between devices).
4	LONG	ZQ_CAL_LENGTH: Indicate short or long ZQ calibration. 0 = SHORT 1 = LONG
0	0x0	ZQ_CAL_CMD: Issues a ZQ calibration command (DDR3 only).

16.7.2.94 EMC_XM2CMDPADCTRL_0

Pad configuration registers from APB_MISC:

XM2CMD Pad Control Register

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2f0 | Read/Write: R/W | Reset: 0x10000200 (0bxxx1xxxxxxxxxxxxxxxx01000000xxx)

Bit	Reset	Description
28	ENABLE	EMC2TMC_CFG_XM2RESET_E_PULLUP: [PMC] Select pullup mode on reset pad 0 = DISABLE 1 = ENABLE
10	NORMAL	CFG_XM2CMD_CAL_SELECT: [PMC] Choose VREF/NORMAL (DQ/DQS) calibration for CMD pads. 0 = NORMAL 1 = VREF
9	ENABLE	EMC2TMC_CFG_XM2RESET_E_DDR3: [PMC] Select pad mode 0 = DISABLE 1 = ENABLE
8	DISABLE	EMC2PMACRO_CFG_XM2CMD_PHASESHIFT: [PMC] 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: [PMC] Select pad mode 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	0x0	EMC2TMC_CFG_XM2CMD_CLK_SEL: [PMC] pad clk_sel (ma bits get this value inverted in LPDDR3 mode)
5	0x0	EMC2TMC_CFG_XM2CMD_E_PREEMP: [PMC] XM2CMD data pins pre-emp enable 0 = DISABLE 1 = ENABLE
4	0x0	EMC2TMC_CFG_XM2CMD_E_BYPASS: [PMC] XM2CMD pins bypass outbound flop enable 0 = DISABLE 1 = ENABLE
3	0x0	EMC2TMC_CFG_XM2CMD_E_PWRD: [PMC] XM2CMD pins pad power-down signal 0 = DISABLE 1 = ENABLE

16.7.2.95 EMC_XM2CMDPADCTRL2_0

Offset: 0x2f4 | Read/Write: R/W | Reset: 0x050c0000 (0b00000101000011000 0xxxxxxxxxxxxxx) | Default: 0x858c0000

Bit	Reset	SW Default	Description
31:28	0x0	0x8	EMC2PMACRO_CFG_XM2CMD_DRVUP_SLWF [PMC]
27:24	0x5	0x5	EMC2PMACRO_CFG_XM2CMD_DRVDN_SLWR [PMC]
23:19	0x1	0x11	EMC2PMACRO_CFG_XM2CMD_DRVUP [PMC]
18:14	0x10	0x10	EMC2PMACRO_CFG_XM2CMD_DRVDN [PMC]

16.7.2.96 EMC_XM2DQSPADCTRL_0

XM2DQS Pad Control Register

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2f8 | Read/Write: R/W | Reset: 0x490c1414 (0b01001001000011000 0x10100xxx10100) | Default: 0x888c1414

Bit	Reset	SW Default	Description
31:28	0x4	0x8	EMC2PMACRO_CFG_XM2DQS_DRVUP_SLWF [PMC]
27:24	0x9	0x8	EMC2PMACRO_CFG_XM2DQS_DRVDN_SLWR [PMC]
23:19	0x1	0x11	EMC2PMACRO_CFG_XM2DQS_DRVUP [PMC]
18:14	0x10	0x10	EMC2PMACRO_CFG_XM2DQS_DRVDN [PMC]
12:8	0x14	0x14	EMC2PMACRO_CFG_XM2DQS_DRVUP_TERM [PMC]
4:0	0x14	0x14	EMC2PMACRO_CFG_XM2DQS_DRVDN_TERM [PMC]

16.7.2.97 EMC_XM2DQSPADCTRL2_0

XM2DQS Pad Control Register

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2fc | Read/Write: R/W | Reset: 0x0000be18 (0bxxxxxxx0xx00000010111110000011000)

Bit	Reset	Description
24	0x0	EMC2TMC_CFG_XM2DQS_QUSE_DLL_BYP: [PMC] 0 = DISABLE 1 = ENABLE
21:20	0x0	EMC2TMC_CFG_XM2DQS_CONFIG: [PMC] IOBRICK rfu pins
19:18	0x0	EMC2TMC_CFG_XM2DQS_RX_DQS_SEL: [PMC] IOBRICK rfu pins
17:16	0x0	EMC2TMC_CFG_XM2DQS_RX_DQ_SEL: [PMC] [0] = select between legacy mode (0) or LSSA mode (1) DQ RX. [1]=RFU
15	0x1	EMC2TMC_CFG_XM2DQS_E_SCHMT_DQ: [PMC] XM2DQ DQ pins Schmitt enable 0 = DISABLE 1 = ENABLE
14	0x0	EMC2TMC_CFG_XM2DQS_E_PWRD: [PMC] XM2DQS pins pad power-down signal 0 = DISABLE 1 = ENABLE
13	0x1	EMC2TMC_CFG_XM2DQS_E_SCHMT_DQS: [PMC] XM2DQS DQS pins Schmitt enable 0 = DISABLE 1 = ENABLE
12	DISABLE	EMC2TMC_CFG_XM2DQS_QUSE_DLI_EN: [PMC] Active low 0 = ENABLE 1 = DISABLE
11	DISABLE	EMC2TMC_CFG_XM2DQS_TXDQS_DLI_EN: [PMC] Active low 0 = ENABLE 1 = DISABLE
10	DISABLE	EMC2TMC_CFG_XM2DQS_TXDQ_DLI_EN: [PMC] Active low 0 = ENABLE 1 = DISABLE
9	DISABLE	EMC2TMC_CFG_XM2DQS_RX_DLI_EN: [PMC] Active low 0 = ENABLE 1 = DISABLE
8	DISABLE	EMC2TMC_CFG_XM2DQS_E_DDR3: [PMC] Select pad mode 0 = DISABLE 1 = ENABLE
7	0x0	EMC2TMC_CFG_XM2DQS_CLKSEL_DQS: [PMC]
6	0x0	EMC2TMC_CFG_XM2DQS_CLKSEL_DQ: [PMC]
5	0x0	EMC2TMC_CFG_XM2DQS_E_VREF_DQ: [PMC] 0 = DISABLE 1 = ENABLE
4	ENABLE	EMC2TMC_CFG_XM2DQS_E_CTT_HIZ_DQS: [PMC] 0 = DISABLE 1 = ENABLE
3	ENABLE	EMC2TMC_CFG_XM2DQS_E_CTT_HIZ_DQ: [PMC] 0 = DISABLE 1 = ENABLE
2	0x0	EMC2TMC_CFG_XM2DQS_E_PREEMP: [PMC] CFG_XM2DQ data pins pre-emp enable 0 = DISABLE 1 = ENABLE
1	0x0	EMC2TMC_CFG_XM2DQS_E_BYPASS: [PMC] CFG_XM2DQ data pins bypass outbound flop enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	0x0	EMC2TMC_CFG_XM2DQS_E_RX_FT_REC: [PMC] 0 = DISABLE 1 = ENABLE

16.7.2.98 EMC_XM2DQPADCTRL_0

CFG_XM2DQ Pad Control Register

These values only used if autocal is disabled

Offset: 0x300 | Read/Write: R/W | Reset: 0x490c2990 (0b01001001000011000010100110011001xxxx) | Default: 0x888c2990

Bit	Reset	SW Default	Description
31:28	0x4	0x8	EMC2PMACRO_CFG_XM2DQ_DRVUP_SLWF [PMC]
27:24	0x9	0x8	EMC2PMACRO_CFG_XM2DQ_DRVDN_SLWR [PMC]
23:19	0x1	0x11	EMC2PMACRO_CFG_XM2DQ_DRVUP [PMC]
18:14	0x10	0x10	EMC2PMACRO_CFG_XM2DQ_DRVDN [PMC]
13:9	0x14	0x14	EMC2PMACRO_CFG_XM2DQ_DRVUP_TERM [PMC]
8:4	0x19	0x19	EMC2PMACRO_CFG_XM2DQ_DRVDN_TERM [PMC]

16.7.2.99 EMC_XM2DQPADCTRL2_0

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x304 | Read/Write: R/W | Reset: 0x00000000 (0bx000x000x000x000xxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:28	0x0	EMC2TMC_CFG_XM2DQ3_DLYIN_TRM: [PMC3] delay trim for byte 3
26:24	0x0	EMC2TMC_CFG_XM2DQ2_DLYIN_TRM: [PMC3] delay trim for byte 2
22:20	0x0	EMC2TMC_CFG_XM2DQ1_DLYIN_TRM: [PMC3] delay trim for byte 1
18:16	0x0	EMC2TMC_CFG_XM2DQ0_DLYIN_TRM: [PMC3] delay trim for byte 0

16.7.2.100 EMC_XM2CLKPADCTRL_0

XM2CLK Pad Control Register

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x308 | Read/Write: R/W | Reset: 0xffffc000 (0b11111111111111111110000000xx000x0)

Bit	Reset	Description
31:28	0xf	EMC2PMACRO_CFG_XM2CLK_DRVUP_SLWF [PMC]

Bit	Reset	Description
27:24	0xf	EMC2PMACRO_CFG_XM2CLK_DRVDN_SLWR [PMC]
23:19	0x1f	EMC2PMACRO_CFG_XM2CLK_DRVUP: [PMC]
18:14	0x1f	EMC2PMACRO_CFG_XM2CLK_DRVDN: [PMC]
7	DISABLE	EMC2TMC_CFG_XM2CLK_E_DDR3: [PMC] Select pad mode 0 = DISABLE 1 = ENABLE
4	DISABLE	EMC2TMC_CFG_XM2CLK_E_PWRD: [PMC] XM2CLK pins pad power-down signal 0 = DISABLE 1 = ENABLE
3	DISABLE	EMC2TMC_CFG_XM2CLK_E_CAL_BYPASS: [PMC] XM2CLK bypass drvdown/up calibration 0 = DISABLE 1 = ENABLE
2	DISABLE	EMC2TMC_CFG_XM2CLK_E_PREEMP: [PMC] pre-emphasis enable 0 = DISABLE 1 = ENABLE
0	DISABLE	EMC2PMACRO_CFG_DYN_PULLS_ON_CLKDIS: [PMC] Enable pull-up on CLKP and pull-down on CLKN when disabling clock. Reserved in case clock startup ramp is too slow (DDR3). 0 = DISABLE 1 = ENABLE

16.7.2.101 EMC_XM2COMPPADCTRL_0

MEM_COMP Pad Control Register

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x30c | Read/Write: R/W | Reset: 0x81f1f008 (0b10000xx11111xxx11111000000001000)

Bit	Reset	Description
31:28	0x8	EMC2TMC_CFG_XM2COMP_PU_VREF_SEL: [PMC]
27	0x0	EMC2TMC_CFG_XM2COMP_PU_VREF_SEL4: [PMC] bit 4 of the PU_VREF_SEL
24:20	0x1f	CFG_XM2COMP_DRVUP: [PMC] used if AUTOCAL is disabled
16:12	0x1f	CFG_XM2COMP_DRVDN: [PMC] used if AUTOCAL is disabled
11	DISABLE	EMC2TMC_CFG_XM2COMP_E_TESTOUT: [PMC] 0 = DISABLE 1 = ENABLE
10	DISABLE	CFG_XM2COMP_VREF_CAL_EN: [PMC] When enabled (=1), generate MCLK calibration cycle. 0 = DISABLE 1 = ENABLE
9	DISABLE	EMC2TMC_CFG_XM2COMP_E_PWRD: [PMC] XM2CLK pins pad power-down signal 0 = DISABLE 1 = ENABLE
8	DISABLE	EMC2TMC_CFG_XM2COMP_E_DDR3: [PMC] Select pad mode 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7:5	0x0	EMC2TMC_CFG_XM2COMP_BIAS_SEL: [PMC]
4:0	0x8	EMC2TMC_CFG_XM2COMP_PD_VREF_SEL: [PMC] VREF_SEL for CK

16.7.2.102 EMC_XM2VTTGENPADCTRL_0

XM2 MISC/VTTGEN Pad Control Register

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x310 | Read/Write: R/W | Reset: 0x07070000 (0bxxxxx111xxxxx111xxxxxxxxxxxxx0x0)

Bit	Reset	Description
26:24	0x7	EMC2TMC_CFG_XM2VTTGEN_DRVUP: [PMC]
18:16	0x7	EMC2TMC_CFG_XM2VTTGEN_DRVDN: [PMC]
2	0x0	EMC2TMC_CFG_XM2VTTGEN_E_DDR3: [PMC] Select pad mode
0	0x0	EMC2TMC_CFG_XM2VTTGEN_SHORT: [PMC]

16.7.2.103 EMC_XM2VTTGENPADCTRL2_0

- E_WEAK_BIAS: enables low current bias mode on VTTGEN cell
- E_NO_VTTGEN: disables VTTGEN altogether. One VTTGEN pad in each MEM section is always enabled (DDR3-only)

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x314 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
31:30	DISABLE	EMC2TMC_CFG_XM2VTTGEN_MEMEXCLK_WEAK_BIAS: [PMC] Enable BIAS mode current mode during DPD -- MEM group excluding CLK 0 = DISABLE 1 = ENABLE 2 = RESERVED
29:28	DISABLE	EMC2TMC_CFG_XM2VTTGEN_MEMCLK_WEAK_BIAS: [PMC] Enable BIAS mode current mode during DPD -- MEMCLK pad group 0 = DISABLE 1 = ENABLE 2 = RESERVED
5:0	0x0	EMC2TMC_CFG_XM2VTTGEN_E_NO_VTTGEN: [PMC] Disable optional VTTGEN pads (1 bit per pad) [0]--ADDR0 [1]--ADDR1 [2]--ADDR2 [3]--DATA1 [4]--DATA2 [5]--DATA3

16.7.2.104 EMC_XM2VTTGENPADCTRL3_0

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x318 | Read/Write: R/W | Reset: 0x013bbbbb (0bxxxxxxx1x01110111011101110111011)

Bit	Reset	Description
24	0x1	EMC2TMC_CFG_XM2VTTGEN_VAUXP_BACKUP_BYTE: [PMC]
22:20	0x3	EMC2TMC_CFG_XM2VTTGEN_VAUXP_LEVEL_BYTE: [PMC]
19	0x1	EMC2TMC_CFG_XM2VTTGEN_VCLAMP_BACKUP_BYTE: [PMC]
18:16	0x3	EMC2TMC_CFG_XM2VTTGEN_VCLAMP_LEVEL_BYTE: [PMC]
15	0x1	EMC2TMC_CFG_XM2VTTGEN_VAUXP_BACKUP_CMD: [PMC]
14:12	0x3	EMC2TMC_CFG_XM2VTTGEN_VAUXP_LEVEL_CMD: [PMC]
11	0x1	EMC2TMC_CFG_XM2VTTGEN_VCLAMP_BACKUP_CMD: [PMC]
10:8	0x3	EMC2TMC_CFG_XM2VTTGEN_VCLAMP_LEVEL_CMD: [PMC]
7	0x1	EMC2TMC_CFG_XM2VTTGEN_VAUXP_BACKUP_CLK: [PMC]
6:4	0x3	EMC2TMC_CFG_XM2VTTGEN_VAUXP_LEVEL_CLK: [PMC]
3	0x1	EMC2TMC_CFG_XM2VTTGEN_VCLAMP_BACKUP_CLK: [PMC]
2:0	0x3	EMC2TMC_CFG_XM2VTTGEN_VCLAMP_LEVEL_CLK: [PMC]

16.7.2.105 EMC_EMCPADEN_0

EMC pad enable register

Writing 0 will disable listed direction

Offset: 0x31c | Read/Write: R/W | Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx11)

Bit	Reset	Description
1	0x1	EMC2PMACRO_CFG_PAD_INPUT_EN: Inputs enable for EMC pads 0 = DISABLE 1 = ENABLE
0	0x1	EMC2PMACRO_CFG_PAD_OUTPUT_EN: Outputs enable for EMC pads 0 = DISABLE 1 = ENABLE

16.7.2.106 EMC_XM2DQSPADCTRL4_0

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x320 | Read/Write: R/W | Reset: 0x00208208 (0bxxxxxxxxx01000x01000x01000x01000)

Bit	Reset	Description
22:18	0x8	EMC2TMC_CFG_XM2DQS_BYTE3_VREF_DQ: [PMC]
16:12	0x8	EMC2TMC_CFG_XM2DQS_BYTE2_VREF_DQ: [PMC]
10:6	0x8	EMC2TMC_CFG_XM2DQS_BYTE1_VREF_DQ: [PMC]
4:0	0x8	EMC2TMC_CFG_XM2DQS_BYTE0_VREF_DQ: [PMC]

16.7.2.107 EMC_SCRATCH0_0

This is a scratch register for general use.

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

Bit	Reset	Description
31:0	0x0	SCRATCH:

16.7.2.108 EMC_DLL_XFORM_DQS0_0

Configure Digital DLL

XFORM_DQSx_MULT and XFORM_DQSx_OFFS are a multiplier and offset, respectively, that can be used to modify the DLL output. Final products may be read out in DQS_TRIMMER_RD register. The default is multiply by 1 to keep the DLL's 1/4 cycle delay. Integer + fractional formats for multiplier and offset:

OFFS is 2's complement format, with bit [0] representing integer

The output of the XFORM is xform_out = (xform_in * MULT + OFFS*16)/ 16

Largest allowed MULT = 0x17

In the case of overflow, output is clamped according to these rules:

- out >= 0x600, xform_out = 0x000
- 0x600 > out >= 0x400, xform_out = 0x3ff

That means clamping works within 50% above the maximum (0x3ff) DLL output or 50% below the minimum (0) DLL output.

To make sure that is always satisfied, conservatively program OFFS so that 0x600 < OFFS < (0x5ff – ((MULT * 0x3ff) >> 4)). The DLL output is effectively overridden by setting XFORM_*_MULT to 0 and programming XFORM_*_OFFS[8:1] to override value.

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

The DQS_CURRENT_TRIM_VAL_BYTE_x registers provide a way to observe the values being used by the trimmers

Offset: 0x328 | Read/Write: R/W | Reset: 0x007ff010 (0bxxxxxxxxxx11111111111xxxxxxx10000)

Bit	Reset	Description
22:12	0x7ff	XFORM_DQS0_OFFS: [PMC]
4:0	0x10	XFORM_DQS0_MULT: [PMC]

16.7.2.109 EMC_DLL_XFORM_DQS1_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x32c | Read/Write: R/W | Reset: 0x007ff010 (0bxxxxxxxxx11111111111xxxxxx10000)

Bit	Reset	Description
22:12	0x7ff	XFORM_DQS1_OFFS: [PMC]
4:0	0x10	XFORM_DQS1_MULT: [PMC]

16.7.2.110 EMC_DLL_XFORM_DQS2_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x330 | Read/Write: R/W | Reset: 0x007ff010 (0bxxxxxxxxx11111111111xxxxxx10000)

Bit	Reset	Description
22:12	0x7ff	XFORM_DQS2_OFFS: [PMC]
4:0	0x10	XFORM_DQS2_MULT: [PMC]

16.7.2.111 EMC_DLL_XFORM_DQS3_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x334 | Read/Write: R/W | Reset: 0x007ff010 (0bxxxxxxxxx11111111111xxxxxx10000)

Bit	Reset	Description
22:12	0x7ff	XFORM_DQS3_OFFS: [PMC]
4:0	0x10	XFORM_DQS3_MULT: [PMC]

16.7.2.112 EMC_DLL_XFORM_DQS4_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x338 | Read/Write: R/W | Reset: 0x007ff010 (0bxxxxxxxxx11111111111xxxxxx10000)

Bit	Reset	Description
22:12	0x7ff	XFORM_DQS4_OFFS: [PMC]
4:0	0x10	XFORM_DQS4_MULT: [PMC]

16.7.2.113 EMC_DLL_XFORM_DQS5_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x33c | Read/Write: R/W | Reset: 0x007ff010 (0bxxxxxxxxx11111111111xxxxxx10000)

Bit	Reset	Description
22:12	0x7ff	XFORM_DQS5_OFFS: [PMC]
4:0	0x10	XFORM_DQS5_MULT: [PMC]

16.7.2.114 EMC_DLL_XFORM_DQS6_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x340 | Read/Write: R/W | Reset: 0x007ff010 (0bxxxxxxxxx11111111111xxxxxx10000)

Bit	Reset	Description
22:12	0x7ff	XFORM_DQS6_OFFS: [PMC]
4:0	0x10	XFORM_DQS6_MULT: [PMC]

16.7.2.115 EMC_DLL_XFORM_DQS7_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x344 | Read/Write: R/W | Reset: 0x007ff010 (0bxxxxxxxxx11111111111xxxxxx10000)

Bit	Reset	Description
22:12	0x7ff	XFORM_DQS7_OFFS: [PMC]
4:0	0x10	XFORM_DQS7_MULT: [PMC]

16.7.2.116 EMC_DLL_XFORM_QUSE0_0

Configure Digital DLL

XFORM_QUSEx_MULT and XFORM_QUSEx_OFFS are a multiplier and offset, respectively, that can be used to modify the DLL output. Final product may be read out in DQS_TRIMMER_RD register. The default is multiply by 1/2, to give 1/8 cycle delay.

OFFS is 2's complement format, with bit [0] representing integer. The output of the XFORM is xform_out = (xform_in * MULT + OFFS*16)/ 16

Largest allowed MULT = 0x17

In the case of overflow, output is clamped according to these rules:

- out >= 0x600, xform_out = 0x000
- 0x600 > out >= 0x400, xform_out = 0x3ff

That means clamping works within 50% above the maximum (0x3ff) DLL output or 50% below the minimum (0) DLL output. To make sure that is always satisfied, conservatively program OFFS so that 0x600 < OFFS < (0x5ff – ((MULT * 0x3ff) >> 4)). The

DLL output is effectively overridden by setting XFORM_*_MULT to 0 and programming XFORM_*_OFFS[8:1] to override value.

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x348 | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxx00000000000xxxxxxx01100)

Bit	Reset	Description
22:12	0x0	XFORM_QUSE0_OFFS: [PMC]
4:0	0xc	XFORM_QUSE0_MULT: [PMC]

16.7.2.117 EMC_DLL_XFORM_QUSE1_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x34c | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxx00000000000xxxxxxx01100)

Bit	Reset	Description
22:12	0x0	XFORM_QUSE1_OFFS: [PMC]
4:0	0xc	XFORM_QUSE1_MULT: [PMC]

16.7.2.118 EMC_DLL_XFORM_QUSE2_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x350 | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxx00000000000xxxxxxx01100)

Bit	Reset	Description
22:12	0x0	XFORM_QUSE2_OFFS: [PMC]
4:0	0xc	XFORM_QUSE2_MULT: [PMC]

16.7.2.119 EMC_DLL_XFORM_QUSE3_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x354 | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxx00000000000xxxxxxx01100)

Bit	Reset	Description
22:12	0x0	XFORM_QUSE3_OFFS: [PMC]
4:0	0xc	XFORM_QUSE3_MULT: [PMC]

16.7.2.120 EMC_DLL_XFORM_QUSE4_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x358 | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxx00000000000xxxxxxx01100)

Bit	Reset	Description
22:12	0x0	XFORM_QUSE4_OFFS: [PMC]
4:0	0xc	XFORM_QUSE4_MULT: [PMC]

16.7.2.121 EMC_DLL_XFORM_QUSE5_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x35c | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxx00000000000xxxxxxx01100)

Bit	Reset	Description
22:12	0x0	XFORM_QUSE5_OFFS: [PMC]
4:0	0xc	XFORM_QUSE5_MULT: [PMC]

16.7.2.122 EMC_DLL_XFORM_QUSE6_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x360 | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxx00000000000xxxxxxx01100)

Bit	Reset	Description
22:12	0x0	XFORM_QUSE6_OFFS: [PMC]
4:0	0xc	XFORM_QUSE6_MULT: [PMC]

16.7.2.123 EMC_DLL_XFORM_QUSE7_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x364 | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxx00000000000xxxxxxx01100)

Bit	Reset	Description
22:12	0x0	XFORM_QUSE7_OFFS: [PMC]
4:0	0xc	XFORM_QUSE7_MULT: [PMC]

16.7.2.124 EMC_DLL_XFORM_DQ0_0

Configure Digital DLL

XFORM_DQx_MULT and XFORM_DQx_OFFS are a multiplier and offset, respectively, that can be used to modify the DLL output. XFORM_DQx_* applies to all DQ/DM bits in each byte x. Final products may be read out in the DQ_TRIMMER_RD

register. The default is multiply by 1, providing 1/4 cycle skew to DQS to keep DLL's 1/4 cycle delay. Integer + fractional formats for multiplier & offset:

OFFS is 2's complement format, with bit [0] representing integer. The output of the XFORM is xform_out = (xform_in * MULT + OFFS*16)/ 16

Largest allowed MULT = 0x17

In the case of overflow, output is clamped according to these rules:

- out >= 0x600, xform_out = 0x000
- 0x600 > out >= 0x400, xform_out = 0x3ff

That means clamping works within 50% above the maximum (0x3ff) DLL output or 50% below the minimum (0) DLL output. To make sure that is always satisfied, conservatively program OFFS so that 0x600 < OFFS < (0x5ff – ((MULT * 0x3ff) >> 4)).

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x368 | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxx00000000000xxxxxxx10000)

Bit	Reset	Description
22:12	0x0	XFORM_TXDQ0_OFFS: [PMC]
4:0	0x10	XFORM_TXDQ0_MULT: [PMC]

16.7.2.125 EMC_DLL_XFORM_DQ1_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x36c | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxx00000000000xxxxxxx10000)

Bit	Reset	Description
22:12	0x0	XFORM_TXDQ1_OFFS: [PMC]
4:0	0x10	XFORM_TXDQ1_MULT: [PMC]

16.7.2.126 EMC_DLL_XFORM_DQ2_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x370 | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxx00000000000xxxxxxx10000)

Bit	Reset	Description
22:12	0x0	XFORM_TXDQ2_OFFS: [PMC]
4:0	0x10	XFORM_TXDQ2_MULT: [PMC]

16.7.2.127 EMC_DLL_XFORM_DQ3_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x374 | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxx00000000000xxxxxx10000)

Bit	Reset	Description
22:12	0x0	XFORM_TXDQ3_OFFS: [PMC]
4:0	0x10	XFORM_TXDQ3_MULT: [PMC]

16.7.2.128 EMC_DLI_RX_TRIM0_0

Configure Digital DLL

Offset: 0x378 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_0
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_0

16.7.2.129 EMC_DLI_RX_TRIM1_0

Configure Digital DLL

Offset: 0x37c | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_1
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_1

16.7.2.130 EMC_DLI_RX_TRIM2_0

Configure Digital DLL

Offset: 0x380 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_2
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_2

16.7.2.131 EMC_DLI_RX_TRIM3_0

Configure Digital DLL

Offset: 0x384 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_3
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_3

16.7.2.132 EMC_DLI_RX_TRIM4_0

Configure Digital DLL

Offset: 0x388 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_4
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_4

16.7.2.133 EMC_DLI_RX_TRIM5_0

Configure Digital DLL

Offset: 0x38c | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_5
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_5

16.7.2.134 EMC_DLI_RX_TRIM6_0

Configure Digital DLL

Offset: 0x390 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_6
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_6

16.7.2.135 EMC_DLI_RX_TRIM7_0

Configure Digital DLL

Offset: 0x394 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_7
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_7

16.7.2.136 EMC_DLI_TX_TRIM0_0

Configure Digital DLL

Offset: 0x398 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	TXDQ_CURRENT_TRIM_VAL_BYTE_0

16.7.2.137 EMC_DLI_TX_TRIM1_0

Configure Digital DLL

Offset: 0x39c | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	TXDQ_CURRENT_TRIM_VAL_BYTE_1

16.7.2.138 EMC_DLI_TX_TRIM2_0

Configure Digital DLL

Offset: 0x3a0 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	TXDQ_CURRENT_TRIM_VAL_BYTE_2

16.7.2.139 EMC_DLI_TX_TRIM3_0

Configure Digital DLL

Offset: 0x3a4 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	TXDQ_CURRENT_TRIM_VAL_BYTE_3

16.7.2.140 EMC_DLI_TRIM_TXDQS0_0

Set DLI TRIM

DLI_TRIM_TXDQSx

Controls trimmer used to skew data byte DQS/DQ output relative to MCK. 0-7 apply to rank 0, 8-15 apply to rank 1 (if dual-rank enabled)

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x3a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS0_DLI: [PMC]

16.7.2.141 EMC_DLI_TRIM_TXDQS1_0

Set DLI TRIM

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x3ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS1_DLI: [PMC]

16.7.2.142 EMC_DLI_TRIM_TXDQS2_0

Set DLI TRIM

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x3b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS2_DLI: [PMC]

16.7.2.143 EMC_DLI_TRIM_TXDQS3_0

Set DLI TRIM

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x3b4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS3_DLI: [PMC]

16.7.2.144 EMC_DLI_TRIM_TXDQS4_0

Set DLI TRIM

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x3b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS4_DLI: [PMC]

16.7.2.145 EMC_DLI_TRIM_TXDQS5_0

Set DLI TRIM

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x3bc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS5_DLI: [PMC]

16.7.2.146 EMC_DLI_TRIM_TXDQS6_0

Set DLI TRIM

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x3c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS6_DLI: [PMC]

16.7.2.147 EMC_DLI_TRIM_TXDQS7_0

Set DLI TRIM

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x3c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS7_DLI: [PMC]

16.7.2.148 EMC_STALL_THEN_EXE_BEFORE_CLKCHANGE_0

Scheme to change controller timing parameters + SDRAM mode timing atomically.

1. Program the EMC controller timing registers (shadowed).
2. Pre-program the SDRAM mode registers.
    a. Write 1 to the STALL_THEN_EXE_BEFORE_CLKCHANGE field, which will prevent further cfg execution until a clock change request has been detected.

    Note: After this step, no register read should happen on the EMC before the clock change (C) happens, otherwise the system will hang. Disable the 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,
        i. Issue a write to SDRAM's MR1 register via EMC MRS register to disable DLL.
        ii. Issue a Self-refresh entry via EMC SELF_REF register.

    Note: If there is 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 config execution until after the actual clock change has happened.
    d. Write to whatever EMC registers you want to be executed after the clock change. Using the same example as (b), you will,
        i. Issue a Self-refresh exit via EMC SELF_REF register.
        ii. Issue writes to SDRAM's MRx register(s) to change read/write latencies and/or enable DLL
        iii. Issue burst refreshes via EMC REF register.

    Note: If there is no need to execute any register access after clock change, simply skip step (d); step (c) is still required.

3. Program the CAR to change either clock source and/or clock divider.

> **Note:** In both sequences above, DYN_SELF_REF must be disabled off before the first step, then it can be restored after the last step.
>
> The interval between two clock change sequences are recommended to be at least 20 µs apart

Writes 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 the clock change request will be acknowledged.

Offset: 0x3c8 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	STALL_THEN_EXE_BEFORE_CLKCHANGE: Writing a one to this bit will stall subsequent register accesses.

16.7.2.149 EMC_STALL_THEN_EXE_AFTER_CLKCHANGE_0

Writes 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 happens, whatever register access follows this register write will be executed until UNSTALL_RW_AFTER_CLKCHANGE is encountered. At that point, normal memory read/write will be allowed to resume.

Offset: 0x3cc | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	STALL_THEN_EXE_AFTER_CLKCHANGE: Writing a one to this bit will stall subsequent register accesses.

16.7.2.150 EMC_UNSTALL_RW_AFTER_CLKCHANGE_0

Writing to this register will unstall memory reads/writes after a clock change. Use in conjunction with STALL_THEN_EXE_AFTER_CLKCHANGE.

Offset: 0x3d0 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	UNSTALL_RW_AFTER_CLKCHANGE: Writing a one to this bit will unstall memory reads/writes after a clock change.

16.7.2.151 EMC_AUTO_CAL_CLK_STATUS_0

EMC Pad Calibration Status

Offset: 0x3d4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:24	X	AUTO_CAL_CLK_PULLDOWN_ADJ: Pull-down code sent to pads for MCLK pad.
20:16	X	AUTO_CAL_CLK_PULLUP_ADJ: Pull-up code sent to pads for MCLK pad.
12:8	X	AUTO_CAL_CLK_PULLDOWN: Pull-down code generated by auto-calibration for MCLK pad.
4:0	X	AUTO_CAL_CLK_PULLUP: Pull-up code generated by auto-calibration for MCLK pad.

16.7.2.152 EMC_SEL_DPD_CTRL_0

Configures Functional SEL_DPD Modes

Offset: 0x3d8 | Read/Write: R/W | Reset: 0x00040000 (0bxxxxxxxxxxxxxx100xxxxxxx0xx0000xx)

Bit	Reset	Description
18:16	0x4	SEL_DPD_DLY: [PMC] number of cycles to wait before asserting SEL_DPD
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
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

16.7.2.153 EMC_PRE_REFRESH_REQ_CNT_0

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x3dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	PRE_REF_REQ_CNT: [PMC] When the refresh counter reach this pre-refresh request count just before a burst refresh will be issued when it reach the {REFRESH,REFRESH_LO}, a pre-refresh request will be asserted to MC to close the banks/pages and flush its pipeline. A 0 value will disable this feature.

16.7.2.154 EMC_DYN_SELF_REF_CONTROL_0

Threshold for Dynamic Self-Refresh Entry

This register controls how dynamic self-refresh entry/exit is handled.

If ODT is enabled, the DSR_PER_DEVICE field must be set to DISABLED.

It is recommended to change the values of ODT_WRITE and DSR_PER_DEVICE with a clock change

To do it outside of a clock change sequence, one has to be aware that DSR enable/disable takes time to take effect. This is the sequence for disabling DSR_PER_DEVICE and enabling ODT outside of clock change:

1. DYN_SELF_REF <= DISABLED (DSR exit)
2. TIMING_UPDATE <= 1 (latch in the shadow value)
3. Read back and discard any EMC register such as INTSTATUS (ensure 1 and 2 have gone through)
4. Wait 2 µs for the last possible self-refresh entry
5. Poll for SDRAM_IN_SELF_REFRESH == 0 (ensure that self-refresh exits)

6. ODT_WRITE <= new value to turn on ODT

   DSR_PER_DEVICE <= DISABLED (turn off per-device DSR)

   DYN_SELF_REF <= ENABLED (re-enable DSR)

7. TIMING_UPDATE <= 1 (latch in the shadow values)

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x3e0 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	DISABLED	DSR_PER_DEVICE: [PMC] controls whether dynamic self-refresh is done on a per-device basis. 0 = DISABLED 1 = ENABLED
15:0	0x0	DSR_THRESHOLD: [PMC] number of idle cycles to wait before allowing dynamic self-refresh entry

16.7.2.155 EMC_TXSRDLL_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Note: If operating in non-DLL mode, this register needs to be updated with the non-DLL timing requirement.

Offset: 0x3e4 | Read/Write: R/W | Reset: 0x000007ff (0bxxxxxxxxxxxxxxxxxxxxx011111111111)

Bit	Reset	Description
11:0	0x7ff	TXSRDLL: [PMC] Cycles between self-refresh exit and first DRAM command requiring a locked DLL. For DDR3 only: READ (and RAP) and synchronous ODT commands. DDR3: 512. LPDDR3: TXSR (not used). Largest allowed value is 0xffe

16.7.2.156 EMC_CCFIFO_ADDR_0

CCFIFO_ADDR: This register contains the address offset of the EMC register that is intended to be executed during the clock change sequence.

> **Note:** Before triggering a clock change sequence from the CAR, you must configure the pre-/post- clock change sequence by writing, multiple times, to the CCFIFO_DATA/CCFIFO_ADDR register pair. After the CCFIFO_ADDR register is written, Hardware will write the register offset and corresponding data into the 32-deep clock change FIFO (CCFIFO).

Offset: 0x3e8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	CCFIFO_ADDR: clock change FIFO address register.

16.7.2.157 EMC_CCFIFO_DATA_0

CCFIFO_DATA: This register contains the 32-bit data of the EMC register that is intended to be written to or pointed at by CCFIFO_ADDR.

Offset: 0x3ec | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	CCFIFO_DATA: clock change FIFO data register.

16.7.2.158 EMC_CCFIFO_STATUS_0

Offset: 0x3f0 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
5:0	X	CCFIFO_COUNT

16.7.2.159 EMC_CDB_CNTL_1_0

CDB Control Register

The following table lists the mapping for the SEL_PI_TAP bits in CDB Control Register 1.

Table 58: SEL_PI_TAP Mapping

CDB SEL_PI_TAP#	EMC Signal	Comment
CDB SEL_PI_TAP15	EMC2PMACRO_SEL_PI_TAP15	data brick 7
CDB SEL_PI_TAP14	EMC2PMACRO_SEL_PI_TAP14	data brick 5
CDB SEL_PI_TAP13	EMC2PMACRO_SEL_PI_TAP13	data brick 6
CDB SEL_PI_TAP12	EMC2PMACRO_SEL_PI_TAP12	data brick 4
CDB SEL_PI_TAP11	EMC2PMACRO_SEL_PI_TAP11	CLKBUF 5 controls channel 1 sdr pins: cke_b0, cke_b1, odt_b0, odt_b1, cs_b0_, cs_b1_
CDB SEL_PI_TAP10	EMC2PMACRO_SEL_PI_TAP10	CLKBUF 4 controls channel 1 ddr/2T pins: ba0, ba1, ba2, a10, a12, a13, a15, a_b3, a_b4
CDB SEL_PI_TAP9	EMC2PMACRO_SEL_PI_TAP9	CLKBUF 3 controls channel 1 ddr/2T pins: a_b5,a11,a14
CDB SEL_PI_TAP8	EMC2PMACRO_SEL_PI_TAP8	MCLK_B
CDB SEL_PI_TAP7	EMC2PMACRO_SEL_PI_TAP7	MCLK
CDB SEL_PI_TAP6	EMC2PMACRO_SEL_PI_TAP6	CLKBUF 2 controls channel 0 sdr pins: cs1_ cs0_, odt1, odt0, cke0, cke1
CDB SEL_PI_TAP5	EMC2PMACRO_SEL_PI_TAP5	CLKBUF 1 controls channel 0 ddr/2T pins: ras_,cas_,we_,rst,a2,a3,a4,a6,a8

CDB SEL_PI_TAP#	EMC Signal	Comment
CDB SEL_PI_TAP4	EMC2PMACRO_SEL_PI_TAP4	CLKBUF 0 controls channel 0 ddr/2T pins: a0,a1,a5,a7,a9
CDB SEL_PI_TAP3	EMC2PMACRO_SEL_PI_TAP3	data byte 3
CDB SEL_PI_TAP2	EMC2PMACRO_SEL_PI_TAP2	data byte 1
CDB SEL_PI_TAP1	EMC2PMACRO_SEL_PI_TAP1	data byte 2
CDB SEL_PI_TAP0	EMC2PMACRO_SEL_PI_TAP0	data byte 0

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x3f4 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000000000000000000000)

Bit	Reset	Description
29:27	0x0	EMC2PMACRO_SEL_PI_TAP9: [PMC]
26:24	0x0	EMC2PMACRO_SEL_PI_TAP8: [PMC]
23:21	0x0	EMC2PMACRO_SEL_PI_TAP7: [PMC]
20:18	0x0	EMC2PMACRO_SEL_PI_TAP6: [PMC]
17:15	0x0	EMC2PMACRO_SEL_PI_TAP5: [PMC]
14:12	0x0	EMC2PMACRO_SEL_PI_TAP4: [PMC]
11:9	0x0	EMC2PMACRO_SEL_PI_TAP3: [PMC]
8:6	0x0	EMC2PMACRO_SEL_PI_TAP2: [PMC]
5:3	0x0	EMC2PMACRO_SEL_PI_TAP1: [PMC]
2:0	0x0	EMC2PMACRO_SEL_PI_TAP0: [PMC]

16.7.2.160 EMC_CDB_CNTL_2_0

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x3f8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EMC2PMACRO_CLKNIGATE_TAP15: [PMC]
30	0x0	EMC2PMACRO_CLKNIGATE_TAP14: [PMC]
29	0x0	EMC2PMACRO_CLKNIGATE_TAP13: [PMC]
28	0x0	EMC2PMACRO_CLKNIGATE_TAP12: [PMC]
27	0x0	EMC2PMACRO_CLKNIGATE_TAP11: [PMC]
26	0x0	EMC2PMACRO_CLKNIGATE_TAP10: [PMC]
25	0x0	EMC2PMACRO_CLKNIGATE_TAP9: [PMC]
24	0x0	EMC2PMACRO_CLKNIGATE_TAP8: [PMC]
23	0x0	EMC2PMACRO_CLKNIGATE_TAP7: [PMC]
22	0x0	EMC2PMACRO_CLKNIGATE_TAP6: [PMC]
21	0x0	EMC2PMACRO_CLKNIGATE_TAP5: [PMC]
20	0x0	EMC2PMACRO_CLKNIGATE_TAP4: [PMC]
19	0x0	EMC2PMACRO_CLKNIGATE_TAP3: [PMC]

Bit	Reset	Description
18	0x0	EMC2PMACRO_CLKNIGATE_TAP2: [PMC]
17	0x0	EMC2PMACRO_CLKNIGATE_TAP1: [PMC]
16	0x0	EMC2PMACRO_CLKNIGATE_TAP0: [PMC]
15	0x0	EMC2PMACRO_CLKPIGATE_TAP15: [PMC]
14	0x0	EMC2PMACRO_CLKPIGATE_TAP14: [PMC]
13	0x0	EMC2PMACRO_CLKPIGATE_TAP13: [PMC]
12	0x0	EMC2PMACRO_CLKPIGATE_TAP12: [PMC]
11	0x0	EMC2PMACRO_CLKPIGATE_TAP11: [PMC]
10	0x0	EMC2PMACRO_CLKPIGATE_TAP10: [PMC]
9	0x0	EMC2PMACRO_CLKPIGATE_TAP9: [PMC]
8	0x0	EMC2PMACRO_CLKPIGATE_TAP8: [PMC]
7	0x0	EMC2PMACRO_CLKPIGATE_TAP7: [PMC]
6	0x0	EMC2PMACRO_CLKPIGATE_TAP6: [PMC]
5	0x0	EMC2PMACRO_CLKPIGATE_TAP5: [PMC]
4	0x0	EMC2PMACRO_CLKPIGATE_TAP4: [PMC]
3	0x0	EMC2PMACRO_CLKPIGATE_TAP3: [PMC]
2	0x0	EMC2PMACRO_CLKPIGATE_TAP2: [PMC]
1	0x0	EMC2PMACRO_CLKPIGATE_TAP1: [PMC]
0	0x0	EMC2PMACRO_CLKPIGATE_TAP0: [PMC]

16.7.2.161 EMC_XM2CLKPADCTRL2_0

XM2CLK Pad Control Register

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x3fc | Read/Write: R/W | Reset: 0x00000000 (0b0xx000000xx00000xx000000xx000000)

Bit	Reset	Description
31	0x0	EMC2PMACRO_CFG_FLIP_CKB_SENSE: [PMC] Flip the m2clk_b pad sense
28:24	0x0	EMC2PMACRO_CFG_PI_CLKB_TRIM: [PMC] pi_clk_trim to control the 32-tap trimmer on the ckb pad macro (pi clock domain)
23	0x0	EMC2PMACRO_CFG_FLIP_CK_SENSE: [PMC] Flip the m2clk pad sense
20:16	0x0	EMC2PMACRO_CFG_PI_CLK_TRIM: [PMC] pi_clk_trim to control the 32-tap trimmer on the ck pad-macro (pi clk domain)
13:8	0x0	EMC2TMC_CFG_XM2CLKB_DLY_TRM_P: [PMC]
5:0	0x0	EMC2TMC_CFG_XM2CLK_DLY_TRM_P: [PMC]

16.7.2.162 EMC_SWIZZLE_RANK0_BYTE_CFG_0

For byte swizzling, each SWZ_RANK*_BYTE*_SEL field indicates which SDRAM byte is mapped/connected to the corresponding byte of the chip.

The SWZ_RANK*_BYTE*_BIT*_SEL fields indicate which SDRAM data bit (0-7) within each byte is mapped/connected to the corresponding bit of the chip.

The Tegra K1 device has 8 bytes per partition without a per rank differentiation for swizzling. The RANK1 register set controls swizzling for the upper 4 byte of data.

Offset: 0x400 | Read/Write: R/W | Reset: 0x00003210 (0bxxxxxxxxxxxxxxxxxxxx11xx10xx01xx00)

Bit	Reset	Description
13:12	0x3	SWZ_RANK0_BYTE3_SEL: [PMC]
9:8	0x2	SWZ_RANK0_BYTE2_SEL: [PMC]
5:4	0x1	SWZ_RANK0_BYTE1_SEL: [PMC]
1:0	0x0	SWZ_RANK0_BYTE0_SEL: [PMC]

16.7.2.163 EMC_SWIZZLE_RANK0_BYTE0_0

Offset: 0x404 | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK0_BYTE0_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK0_BYTE0_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK0_BYTE0_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK0_BYTE0_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK0_BYTE0_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK0_BYTE0_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK0_BYTE0_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK0_BYTE0_BIT0_SEL: [PMC]

16.7.2.164 EMC_SWIZZLE_RANK0_BYTE1_0

Offset: 0x408 | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK0_BYTE1_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK0_BYTE1_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK0_BYTE1_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK0_BYTE1_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK0_BYTE1_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK0_BYTE1_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK0_BYTE1_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK0_BYTE1_BIT0_SEL: [PMC]

16.7.2.165 EMC_SWIZZLE_RANK0_BYTE2_0

Offset: 0x40c | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK0_BYTE2_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK0_BYTE2_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK0_BYTE2_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK0_BYTE2_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK0_BYTE2_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK0_BYTE2_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK0_BYTE2_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK0_BYTE2_BIT0_SEL: [PMC]

16.7.2.166 EMC_SWIZZLE_RANK0_BYTE3_0

Offset: 0x410 | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK0_BYTE3_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK0_BYTE3_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK0_BYTE3_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK0_BYTE3_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK0_BYTE3_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK0_BYTE3_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK0_BYTE3_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK0_BYTE3_BIT0_SEL: [PMC]

16.7.2.167 EMC_SWIZZLE_RANK1_BYTE_CFG_0

Offset: 0x414 | Read/Write: R/W | Reset: 0x00003210 (0bxxxxxxxxxxxxxxxxxxxx11xx10xx01xx00)

Bit	Reset	Description
13:12	0x3	SWZ_RANK1_BYTE3_SEL: [PMC]
9:8	0x2	SWZ_RANK1_BYTE2_SEL: [PMC]
5:4	0x1	SWZ_RANK1_BYTE1_SEL: [PMC]
1:0	0x0	SWZ_RANK1_BYTE0_SEL: [PMC]

16.7.2.168 EMC_SWIZZLE_RANK1_BYTE0_0

Offset: 0x418 | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK1_BYTE0_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK1_BYTE0_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK1_BYTE0_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK1_BYTE0_BIT4_SEL: [PMC]

Bit	Reset	Description
14:12	0x3	SWZ_RANK1_BYTE0_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK1_BYTE0_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK1_BYTE0_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK1_BYTE0_BIT0_SEL: [PMC]

16.7.2.169 EMC_SWIZZLE_RANK1_BYTE1_0

Offset: 0x41c | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK1_BYTE1_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK1_BYTE1_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK1_BYTE1_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK1_BYTE1_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK1_BYTE1_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK1_BYTE1_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK1_BYTE1_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK1_BYTE1_BIT0_SEL: [PMC]

16.7.2.170 EMC_SWIZZLE_RANK1_BYTE2_0

Offset: 0x420 | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK1_BYTE2_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK1_BYTE2_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK1_BYTE2_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK1_BYTE2_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK1_BYTE2_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK1_BYTE2_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK1_BYTE2_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK1_BYTE2_BIT0_SEL: [PMC]

16.7.2.171 EMC_SWIZZLE_RANK1_BYTE3_0

Offset: 0x424 | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK1_BYTE3_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK1_BYTE3_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK1_BYTE3_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK1_BYTE3_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK1_BYTE3_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK1_BYTE3_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK1_BYTE3_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK1_BYTE3_BIT0_SEL: [PMC]

16.7.2.172 EMC_CA_TRAINING_START_0

Procedure for LPDDR3 CA Training

(1a) Configure data swizzling (if needed) through the DATA_SWIZZLE_RANK[0,1]_[0:7] registers.

(1b) Enable EMC2PMACRO_CFG_BYPASS_ADDRPIPE if needed. There's no need to enable EMC2PMACRO_CFG_BYPASS_DATAPIPE* for CA training purposes.

(1c) Follow the procedure on optimal trim setting into the CA bus clock trimmer below.

(2) Configure the CA training engine through the CA_TRAINING_[TIMING_CNTL1,TIMING_CNTL2, CA_LEAD_IN, CA, and CA_LEAD_OUT] registers.

(3) Software configures the CA training engine to use MR41 mode (through the CA_TRAINING_CFG register).

(4) Software starts the CA training engine (in the CA_TRAINING_START register) and waits for the CA training to finish (CA_TRAINING_BUSY).

(5) Software saves the result from the CA_TRAINING_RESULT* registers.

(6) Software repeats steps 3-5 but configures for MR48 mode.

(7) Software combines the results from MR41/MR48 modes to come up with a single trim setting and programs the EMC2PMACRO_CFG_XM2CMD_DIGTRIM field.

(8) Software issues MR42 to rank0.

(9) If desired, software can repeat steps 3-8 for rank1.

(10) Software combines the results from MR41/MR48 modes (if desired, results for rank1 as well) to come up with a single trim setting and programs the EMC2PMACRO_CFG_XM2CMD_DIGTRIM field

(A) Put CLKBUF0/CLKBUF2 into trimmer mode by setting EMC2TMC_CFG_XM2CMD_CLKBUF[0,2]_CONFIG = 001.

(B) Put XFORM_ADDR[0,2]_MULT field to 5'b0.

(C) Program bits [7:2] in the XFORM_ADDR[0,2]_OFFS field (11 bits) with the optimal trim result. The other XFORM_ADDR[0,2]_OFFS bits should remain at 0.

(D) Write a 1 into the TIMING_UPDATE field to load the above fields from shadow to active.

Below is what should be programmed into the CA_TRAINING_TIMING_CONTROL* registers.

CACKEL = tCACKEL + 15 clks

CAENT = tCAENT + 3 clks

CACD = tCACD + 5 clks

CACKEH = tCACKEH – 1 clk

CAEXT = tCAEXT – 1 clk

CA_DQ_RDV = CACD

Offset: 0x428 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	CA_TRAIN_START: 1 = start training.

16.7.2.173 EMC_CA_TRAINING_BUSY_0

Offset: 0x42c | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	CA_TRAIN_BUSY: 1 = training is in progress.

16.7.2.174 EMC_CA_TRAINING_CFG_0

Offset: 0x430 | Read/Write: R/W | Reset: 0x0000001f (0bx000xx0000000000x0000000x0011111)

Bit	Reset	Description
30	RANK0	CA_RANK: Indicates which rank to do training on. 0 = RANK0 1 = RANK1
29	0x0	CA_DRAM_X32: 1 = Using x32 DRAM, 0 = using x16 DRAM.
28	MR41	CA_MR48: 0 = training with MR41 mode 1 = training with MR48 mode. 0 = MR41 1 = MR48
25:16	0x0	CA_MRW_CA: MR41 = 0x290, MR48 = 0x300.
14:8	0x0	CA_START_TRIM_VAL: Indicates the starting delay trim value to use for CA training.
6:0	0x1f	CA_END_TRIM_VAL: Indicates the ending delay trim value to use for CA training.

16.7.2.175 EMC_CA_TRAINING_TIMING_CNTL1_0

Offset: 0x434 | Read/Write: R/W | Reset: 0x1f7df7df (0bxxx11111x11111x11111x11111x11111)

Bit	Reset	Description
28:24	0x1f	CAEXT: tCAEXT parameter.
22:18	0x1f	CACKEH: tCACKEH parameter.
16:12	0x1f	CACD: tCACD parameter.
10:6	0x1f	CAENT: tCAENT parameter.
4:0	0x1f	CACKEL: tCACKEL parameter.

16.7.2.176 EMC_CA_TRAINING_TIMING_CNTL2_0

Offset: 0x438 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx011111)

Bit	Reset	Description
5:0	0x1f	CA_DQ_RDV: number of clocks to delay after CS is asserted before comparing CA/DQ. Minimum is 5. 40 = MAX

16.7.2.177 EMC_CA_TRAINING_CA_LEAD_IN_0

Offset: 0x43c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx0000000000xxxxxx0000000000)

Bit	Reset	Description
25:16	0x0	CA_LEAD_IN_RISE: CA rise value for the clock before CS is active.
9:0	0x0	CA_LEAD_IN_FALL: CA fall value for the clock before CS is active.

16.7.2.178 EMC_CA_TRAINING_CA_0

Offset: 0x440 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx0000000000xxxxxx0000000000)

Bit	Reset	Description
25:16	0x0	CA_RISE: CA rise value for the clock CS is active.
9:0	0x0	CA_FALL: CA fall value for the clock CS is active.

16.7.2.179 EMC_CA_TRAINING_CA_LEAD_OUT_0

Offset: 0x444 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx0000000000xxxxxx0000000000)

Bit	Reset	Description
25:16	0x0	CA_LEAD_OUT_RISE: CA rise value for the clock after CS is active.
9:0	0x0	CA_LEAD_OUT_FALL: CA fall value for the clock after CS is active.

16.7.2.180 EMC_CA_TRAINING_RESULT1_0

Offset: 0x448 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CA_TRIM_RESULT_31_0: Indicates which trim setting(s) pass the CA-to-DQ comparison.

16.7.2.181 EMC_CA_TRAINING_RESULT2_0

Offset: 0x44c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CA_TRIM_RESULT_63_32: Indicates which trim setting(s) pass the CA-to-DQ comparison.

16.7.2.182 EMC_CA_TRAINING_RESULT3_0

Offset: 0x450 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CA_TRIM_RESULT_95_64: Indicates which trim setting(s) pass the CA-to-DQ comparison.

16.7.2.183 EMC_CA_TRAINING_RESULT4_0

Offset: 0x454 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CA_TRIM_RESULT_127_96: Indicates which trim setting(s) pass the CA-to-DQ comparison.

16.7.2.184 EMC_AUTO_CAL_CONFIG2_0

Auto-Calibration Settings for EMC Pads

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x458 | Read/Write: R/W | Reset: 0x00000000 (0bxxx00000xxx00000xxx00000xxx00000)

Bit	Reset	Description
28:24	0x0	AUTO_CAL_CNTL_PD_OFFSET: [PMC] 2's complement offset for CS/ODT/CKE pull-down value
20:16	0x0	AUTO_CAL_CNTL_PU_OFFSET: [PMC] 2's complement offset for CS/ODT/CKE pull-up value
12:8	0x0	AUTO_CAL_CMD_PD_OFFSET: [PMC] 2's complement offset for ADDR/RAS/CAS/WE pull-down value
4:0	0x0	AUTO_CAL_CMD_PU_OFFSET: [PMC] 2's complement offset for ADDR/RAS/CAS/WE pull-up value

16.7.2.185 EMC_AUTO_CAL_CONFIG3_0

Auto-Calibration Settings for EMC Pads

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x45c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx00000xxx00000)

Bit	Reset	Description
12:8	0x0	AUTO_CAL_CLK_PD_OFFSET: [PMC] 2's complement offset for CLK pull-down value
4:0	0x0	AUTO_CAL_CLK_PU_OFFSET: [PMC] 2's complement offset for CLK pull-up value

16.7.2.186 EMC_AUTO_CAL_STATUS2_0

EMC Pad Calibration Status

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

Bit	Reset	Description
28:24	X	AUTO_CAL_CNTL_PULLDOWN_ADJ: Pullup/down code sent to pads
20:16	X	AUTO_CAL_CNTL_PULLUP_ADJ: Pullup/down code sent to pads
12:8	X	AUTO_CAL_CMD_PULLDOWN_ADJ: Pullup/down code sent to pads.
4:0	X	AUTO_CAL_CMD_PULLUP_ADJ: Pullup/down code sent to pads

16.7.2.187 EMC_IBDLY_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x468 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00001)

Bit	Reset	Description
4:0	0x1	IBDLY: [PMC] tells the chip when to change IBDLY for DQ/DQS.

16.7.2.188 EMC_DLL_XFORM_ADDR0_0

Configure Digital DLL for CMD/ADDR Group 0

XFORM_ADDRx_MULT and XFORM_ADDRx_OFFS are a multiplier and offset, respectively, that can be used to modify the DLL output. Final products may be read out in the ADDR_TRIMMER_RD register. The default is multiply by 1, to keep the DLL's 1/4 cycle delay. Integer + fractional formats for multiplier and offset:

OFFS is 2's complement format, with bit [0] representing integer

The output of the XFORM is xform_out = (xform_in * MULT + OFFS*16)/16

The largest allowed MULT = 0x17

In the case of overflow, the output is clamped according to these rules:

out >= 0x600, xform_out = 0x000

0x600 > out >= 0x400, xform_out = 0x3ff

That means clamping works within 50% above the maximum (0x3ff) DLL output or 50% below the minimum (0) DLL output. To make sure that is always satisfied, conservatively program OFFS so that 0x600 < OFFS < (0x5ff - ((MULT * 0x3ff) >> 4))

The DLL output is effectively overridden by setting XFORM_*_MULT to 0 and programming XFORM_*_OFFS[8:1] to override the value

This register is shadowed: see usage note at the top of Section 16.7.2

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

The ADDR(0|1|2)_CURRENT_TRIM_VAL registers provide a way to observe the values being used by the trimmers.

CLKBUF assignments for XFORM_ADDR as well as DLI_DDR_TRIM are:

- XFORM_ADDR5 controls channel 1 sdr pins: cke_b0, cke_b1, odt_b0, odt_b1, cs_b0_, cs_b1_
- XFORM_ADDR4 controls channel 1 ddr/2T pins: ba0, ba1, ba2, a10, a12, a13, a15, a_b3, a_b4
- XFORM_ADDR3 controls channel 1 ddr/2T pins: a_b5, a11, a14
- XFORM_ADDR2 controls channel 0 sdr pins: cs1_ cs0_, odt1, odt0, cke0, cke1
- XFORM_ADDR1 controls channel 0 ddr/2T pins: ras_,cas_,we_,rst,a2,a3, a4, a6, a8
- XFORM_ADDR0 controls channel 0 ddr/2T pins: a0, a1, a5, a7, a9

Offset: 0x46c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx00000000000xxxxxxx00000)

Bit	Reset	Description
22:12	0x0	XFORM_ADDR0_OFFS: [PMC]
4:0	0x0	XFORM_ADDR0_MULT: [PMC]

16.7.2.189 EMC_DLL_XFORM_ADDR1_0

Configure Digital DLL for CMD/ADDR Group 1

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x470 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx00000000000xxxxxxx00000)

Bit	Reset	Description
22:12	0x0	XFORM_ADDR1_OFFS: [PMC]
4:0	0x0	XFORM_ADDR1_MULT: [PMC]

16.7.2.190 EMC_DLL_XFORM_ADDR2_0

Configure Digital DLL for CMD/ADDR Group 2

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x474 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx00000000000xxxxxxx00000)

Bit	Reset	Description
22:12	0x0	XFORM_ADDR2_OFFS: [PMC]
4:0	0x0	XFORM_ADDR2_MULT: [PMC]

16.7.2.191 EMC_DLI_ADDR_TRIM_0

Configure Digital DLL

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

Bit	Reset	Description
29:20	X	ADDR2_CURRENT_TRIM_VAL
19:10	X	ADDR1_CURRENT_TRIM_VAL
9:0	X	ADDR0_CURRENT_TRIM_VAL

16.7.2.192 EMC_DSR_VTTGEN_DRV_0

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x47c | Read/Write: R/W | Reset: 0x0707003f (0bxxxxx111xxxxx111xxxxxxxxx111111)

Bit	Reset	Description
26:24	0x7	DSR_VTTGEN_DRVUP: [PMC] Indicates the VTTGEN VCLAMP regulator impedance to use during DSR.
18:16	0x7	DSR_VTTGEN_DRVDN: [PMC] Indicates the VTTGEN VAUXP regulator impedance to use during DSR.
5:0	0x3f	DSR_VTTGEN_E_NO_VTTGEN: [PMC] Disables optional VTTGEN pads (1 bit per pad) [0]--ADDR0, [1]--ADDR1, [2]--ADDR2, [3]--DATA1, [4]--DATA2, [5]--DATA3

16.7.2.193 EMC_TXDSRVTTGEN_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x480 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	TXDSRVTTGEN: [PMC] Cycles to wait from DSR exit (VTTGEN drive normal), to when a DRAM transaction is allow to start.

16.7.2.194 EMC_XM2CMDPADCTRL4_0

XM2CMD Pad Control Register 4

Controls related to NI-PI handoff in pad macros.

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: 0x484 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx0x0x0x0x0x0)

Bit	Reset	Description
10	0x0	EMC2PMACRO_CFG_2T_PM_PHASE_SHIFT: [PMC] when set to 1, enables 1/2 clocks of delay on OB 2T signals
8	0x0	EMC2PMACRO_CFG_2T_PM2PAD_FLOP_BYPASS: [PMC] when set to 1, enables flop between NI-PI handoff structure and pad for OB 2T
6	0x0	EMC2PMACRO_CFG_1T_PM_PHASE_SHIFT: [PMC] when set to 1, enables 1/2 clocks of delay on OB 1T signals
4	0x0	EMC2PMACRO_CFG_1T_PM2PAD_FLOP_BYPASS: [PMC] when set to 1, enables flop between NI-PI handoff structure and pad for OB 1T
2	0x0	EMC2PMACRO_CFG_DAT_PM_PHASE_SHIFT: [PMC] when set to 1, enables 1/2 clocks of delay on OB data signals
0	0x0	EMC2PMACRO_CFG_DAT_PM2PAD_FLOP_BYPASS: [PMC] when set to 1, enables flop between DQ/DQS NI-PI handoff structure and pad

16.7.2.195 EMC_XM2CMDPADCTRL5_0

This register is shadowed: see usage note at the top of Section 16.7.2

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: 0x488 | Read/Write: R/W | Reset: 0x00111111 (0bxxxxxxxxxx001x001x001x001x001x001)

Bit	Reset	Description
22:20	0x1	EMC2TMC_CFG_XM2CMD_CLKBUF5_CONFIG: [PMC] CONFIG bit setting of CLKBUF5 pad (000 == DLCELL mode, 001 == TRIM/PDL mode)
18:16	0x1	EMC2TMC_CFG_XM2CMD_CLKBUF4_CONFIG: [PMC] CONFIG bit setting of CLKBUF4 pad (000 == DLCELL mode, 001 == TRIM/PDL mode)
14:12	0x1	EMC2TMC_CFG_XM2CMD_CLKBUF3_CONFIG: [PMC] CONFIG bit setting of CLKBUF3 pad (000 == DLCELL mode, 001 == TRIM/PDL mode)
10:8	0x1	EMC2TMC_CFG_XM2CMD_CLKBUF2_CONFIG: [PMC] CONFIG bit setting of CLKBUF2 pad (000 == DLCELL mode, 001 == TRIM/PDL mode)
6:4	0x1	EMC2TMC_CFG_XM2CMD_CLKBUF1_CONFIG: [PMC] CONFIG bit setting of CLKBUF1 pad (000 == DLCELL mode, 001 == TRIM/PDL mode)
2:0	0x1	EMC2TMC_CFG_XM2CMD_CLKBUF0_CONFIG: [PMC] CONFIG bit setting of CLKBUF0 pad (000 == DLCELL mode, 001 == TRIM/PDL mode)

16.7.2.196 EMC_CFG_3_0

EMC Configuration

Boot requirements:

- 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: 0x49c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx000x000)

Bit	Reset	Description
6:4	0x0	MRR_BYTESEL_X16: If using 2 x16 DRAM on a single CS to form 32-bit wide data, indicates to which byte lane the second DRAM's byte 0 is connected.
2:0	0x0	MRR_BYTESEL: Indicates which AP byte lane is connected to DRAM byte 0 (over which MRR data is returned).

16.7.2.197 EMC_DLL_XFORM_DQS8_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x4a0 | Read/Write: R/W | Reset: 0x007ff010 (0bxxxxxxxxx11111111111xxxxxxx10000)

Bit	Reset	Description
22:12	0x7ff	XFORM_DQS8_OFFS: [PMC]
4:0	0x10	XFORM_DQS8_MULT: [PMC]

16.7.2.198 EMC_DLL_XFORM_DQS9_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x4a4 | Read/Write: R/W | Reset: 0x007ff010 (0bxxxxxxxxx11111111111xxxxxxx10000)

Bit	Reset	Description
22:12	0x7ff	XFORM_DQS9_OFFS: [PMC]
4:0	0x10	XFORM_DQS9_MULT: [PMC]

16.7.2.199 EMC_DLL_XFORM_DQS10_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x4a8 | Read/Write: R/W | Reset: 0x007ff010 (0bxxxxxxxxx11111111111xxxxxxx10000)

Bit	Reset	Description
22:12	0x7ff	XFORM_DQS10_OFFS: [PMC]
4:0	0x10	XFORM_DQS10_MULT: [PMC]

16.7.2.200 EMC_DLL_XFORM_DQS11_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x4ac | Read/Write: R/W | Reset: 0x007ff010 (0bxxxxxxxxx11111111111xxxxxxx10000)

Bit	Reset	Description
22:12	0x7ff	XFORM_DQS11_OFFS: [PMC]
4:0	0x10	XFORM_DQS11_MULT: [PMC]

16.7.2.201 EMC_DLL_XFORM_DQS12_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x4b0 | Read/Write: R/W | Reset: 0x007ff010 (0bxxxxxxxxx11111111111xxxxxxx10000)

Bit	Reset	Description
22:12	0x7ff	XFORM_DQS12_OFFS: [PMC]
4:0	0x10	XFORM_DQS12_MULT: [PMC]

16.7.2.202 EMC_DLL_XFORM_DQS13_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x4b4 | Read/Write: R/W | Reset: 0x007ff010 (0bxxxxxxxxx11111111111xxxxxxx10000)

Bit	Reset	Description
22:12	0x7ff	XFORM_DQS13_OFFS: [PMC]
4:0	0x10	XFORM_DQS13_MULT: [PMC]

16.7.2.203 EMC_DLL_XFORM_DQS14_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x4b8 | Read/Write: R/W | Reset: 0x007ff010 (0bxxxxxxxxx11111111111xxxxxxx10000)

Bit	Reset	Description
22:12	0x7ff	XFORM_DQS14_OFFS: [PMC]
4:0	0x10	XFORM_DQS14_MULT: [PMC]

16.7.2.204 EMC_DLL_XFORM_DQS15_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x4bc | Read/Write: R/W | Reset: 0x007ff010 (0bxxxxxxxxx11111111111xxxxxxx10000)

Bit	Reset	Description
22:12	0x7ff	XFORM_DQS15_OFFS: [PMC]
4:0	0x10	XFORM_DQS15_MULT: [PMC]

16.7.2.205 EMC_DLL_XFORM_QUSE8_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x4c0 | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxx000000000000000xxx01100)

Bit	Reset	Description
22:8	0x0	XFORM_QUSE8_OFFS:[PMC]
4:0	0xc	XFORM_QUSE8_MULT:[PMC]

16.7.2.206 EMC_DLL_XFORM_QUSE9_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x4c4 | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxx000000000000000xxx01100)

Bit	Reset	Description
22:8	0x0	XFORM_QUSE9_OFFS:[PMC]
4:0	0xc	XFORM_QUSE9_MULT:[PMC]

16.7.2.207 EMC_DLL_XFORM_QUSE10_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x4c8 | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxx000000000000000xxx01100)

Bit	Reset	Description
22:8	0x0	XFORM_QUSE10_OFFS:[PMC]
4:0	0xc	XFORM_QUSE10_MULT:[PMC]

16.7.2.208 EMC_DLL_XFORM_QUSE11_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x4cc | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxx000000000000000xxx01100)

Bit	Reset	Description
22:8	0x0	XFORM_QUSE11_OFFS:[PMC]
4:0	0xc	XFORM_QUSE11_MULT:[PMC]

16.7.2.209 EMC_DLL_XFORM_QUSE12_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x4d0 | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxx000000000000000xxx01100)

Bit	Reset	Description
22:8	0x0	XFORM_QUSE12_OFFS:[PMC]
4:0	0xc	XFORM_QUSE12_MULT:[PMC]

16.7.2.210 EMC_DLL_XFORM_QUSE13_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x4d4 | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxx000000000000000xxx01100)

Bit	Reset	Description
22:8	0x0	XFORM_QUSE13_OFFS:[PMC]
4:0	0xc	XFORM_QUSE13_MULT:[PMC]

16.7.2.211 EMC_DLL_XFORM_QUSE14_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x4d8 | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxx000000000000000xxx01100)

Bit	Reset	Description
22:8	0x0	XFORM_QUSE14_OFFS:[PMC]
4:0	0xc	XFORM_QUSE14_MULT:[PMC]

16.7.2.212 EMC_DLL_XFORM_QUSE15_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x4dc | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxx000000000000000xxx01100)

Bit	Reset	Description
22:8	0x0	XFORM_QUSE15_OFFS:[PMC]
4:0	0xc	XFORM_QUSE15_MULT:[PMC]

16.7.2.213 EMC_DLL_XFORM_DQ4_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x4e0 | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxx000000000000000xxx10000)

Bit	Reset	Description
22:8	0x0	XFORM_TXDQ4_OFFS:[PMC]
4:0	0x10	XFORM_TXDQ4_MULT:[PMC]

16.7.2.214 EMC_DLL_XFORM_DQ5_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x4e4 | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxx000000000000000xxx10000)

Bit	Reset	Description
22:8	0x0	XFORM_TXDQ5_OFFS:[PMC]
4:0	0x10	XFORM_TXDQ5_MULT:[PMC]

16.7.2.215 EMC_DLL_XFORM_DQ6_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x4e8 | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxx000000000000000xxx10000)

Bit	Reset	Description
22:8	0x0	XFORM_TXDQ6_OFFS:[PMC]
4:0	0x10	XFORM_TXDQ6_MULT:[PMC]

16.7.2.216 EMC_DLL_XFORM_DQ7_0

Configure Digital DLL

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x4ec | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxx000000000000000xxx10000)

Bit	Reset	Description
22:8	0x0	XFORM_TXDQ7_OFFS:[PMC]
4:0	0x10	XFORM_TXDQ7_MULT:[PMC]

16.7.2.217 EMC_DLI_RX_TRIM8_0

Configure Digital DLL

Offset: 0x4f0 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_8
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_8

16.7.2.218 EMC_DLI_RX_TRIM9_0

Configure Digital DLL

Offset: 0x4f4 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_9
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_9

16.7.2.219 EMC_DLI_RX_TRIM10_0

Configure Digital DLL

Offset: 0x4f8 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_10
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_10

16.7.2.220 EMC_DLI_RX_TRIM11_0

Configure Digital DLL

Offset: 0x4fc | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_11
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_11

16.7.2.221 EMC_DLI_RX_TRIM12_0

Configure Digital DLL

Offset: 0x500 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_12
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_12

16.7.2.222 EMC_DLI_RX_TRIM13_0

Configure Digital DLL

Offset: 0x504 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_13
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_13

16.7.2.223 EMC_DLI_RX_TRIM14_0

Configure Digital DLL

Offset: 0x508 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_14
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_14

16.7.2.224 EMC_DLI_RX_TRIM15_0

Configure Digital DLL

Offset: 0x50c | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_15
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_15

16.7.2.225 EMC_DLI_TX_TRIM4_0

Configure Digital DLL

Offset: 0x510 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	TXDQ_CURRENT_TRIM_VAL_BYTE_4

16.7.2.226 EMC_DLI_TX_TRIM5_0

Configure Digital DLL

Offset: 0x514 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	TXDQ_CURRENT_TRIM_VAL_BYTE_5

16.7.2.227 EMC_DLI_TX_TRIM6_0

Configure Digital DLL

Offset: 0x518 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	TXDQ_CURRENT_TRIM_VAL_BYTE_6

16.7.2.228 EMC_DLI_TX_TRIM7_0

Configure Digital DLL

Offset: 0x51c | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	TXDQ_CURRENT_TRIM_VAL_BYTE_7

16.7.2.229 EMC_DLI_TRIM_TXDQS8_0

Set DLI TRIM

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x520 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS8_DLI:[PMC]

16.7.2.230 EMC_DLI_TRIM_TXDQS9_0

Set DLI TRIM

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x524 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS9_DLI:[PMC]

16.7.2.231 EMC_DLI_TRIM_TXDQS10_0

Set DLI TRIM

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x528 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS10_DLI:[PMC]

16.7.2.232 EMC_DLI_TRIM_TXDQS11_0

Set DLI TRIM

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x52c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS11_DLI:[PMC]

16.7.2.233 EMC_DLI_TRIM_TXDQS12_0

Set DLI TRIM

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x530 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS12_DLI:[PMC]

16.7.2.234 EMC_DLI_TRIM_TXDQS13_0

Set DLI TRIM

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x534 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS13_DLI:[PMC]

16.7.2.235 EMC_DLI_TRIM_TXDQS14_0

Set DLI TRIM

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x538 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS14_DLI:[PMC]

16.7.2.236 EMC_DLI_TRIM_TXDQS15_0

Set DLI TRIM

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x53c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS15_DLI:[PMC]

16.7.2.237 EMC_CDB_CNTL_3_0

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x540 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx000000000000000000)

Bit	Reset	Description
17:15	0x0	EMC2PMACRO_SEL_PI_TAP15: [PMC]
14:12	0x0	EMC2PMACRO_SEL_PI_TAP14: [PMC]
11:9	0x0	EMC2PMACRO_SEL_PI_TAP13: [PMC]
8:6	0x0	EMC2PMACRO_SEL_PI_TAP12: [PMC]
5:3	0x0	EMC2PMACRO_SEL_PI_TAP11: [PMC]
2:0	0x0	EMC2PMACRO_SEL_PI_TAP10: [PMC]

16.7.2.238 EMC_XM2DQSPADCTRL5_0

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x544 | Read/Write: R/W | Reset: 0x00208208 (0bxxxxxxxxx01000x01000x01000x01000)

Bit	Reset	Description
22:18	0x8	EMC2TMC_CFG_XM2DQS_BYTE7_VREF_DQ: [PMC]

Bit	Reset	Description
16:12	0x8	EMC2TMC_CFG_XM2DQS_BYTE6_VREF_DQ: [PMC]
10:6	0x8	EMC2TMC_CFG_XM2DQS_BYTE5_VREF_DQ: [PMC]
4:0	0x8	EMC2TMC_CFG_XM2DQS_BYTE4_VREF_DQ: [PMC]

16.7.2.239 EMC_XM2DQSPADCTRL6_0

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x548 | Read/Write: R/W | Reset: 0x20820800 (0bx01000x01000x01000x01000xxxxxxxx)

Bit	Reset	Description
30:26	0x8	EMC2PMACRO_CFG_XM2DQS_BYTE7_VREF_DQS: [PMC]
24:20	0x8	EMC2PMACRO_CFG_XM2DQS_BYTE6_VREF_DQS: [PMC]
18:14	0x8	EMC2PMACRO_CFG_XM2DQS_BYTE5_VREF_DQS: [PMC]
12:8	0x8	EMC2PMACRO_CFG_XM2DQS_BYTE4_VREF_DQS: [PMC]

16.7.2.240 EMC_XM2DQPADCTRL3_0

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x54c | Read/Write: R/W | Reset: 0x00000000 (0bx000x000x000x000xxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:28	0x0	EMC2TMC_CFG_XM2DQ7_DLYIN_TRM: [PMC] delay trim for byte 3
26:24	0x0	EMC2TMC_CFG_XM2DQ6_DLYIN_TRM: [PMC] delay trim for byte 2
22:20	0x0	EMC2TMC_CFG_XM2DQ5_DLYIN_TRM: [PMC] delay trim for byte 1
18:16	0x0	EMC2TMC_CFG_XM2DQ4_DLYIN_TRM: [PMC] delay trim for byte 0

16.7.2.241 EMC_DLL_XFORM_ADDR3_0

Configure Digital DLL for CMD/ADDR group 3

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x550 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx00000000000xxxxxxx00000)

Bit	Reset	Description
22:12	0x0	XFORM_ADDR3_OFFS: [PMC]
4:0	0x0	XFORM_ADDR3_MULT: [PMC]

16.7.2.242 EMC_DLL_XFORM_ADDR4_0

Configure Digital DLL for CMD/ADDR group 4

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x554 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx00000000000xxxxxxx00000)

Bit	Reset	Description
22:12	0x0	XFORM_ADDR4_OFFS: [PMC]
4:0	0x0	XFORM_ADDR4_MULT: [PMC]

16.7.2.243 EMC_DLL_XFORM_ADDR5_0

Configure Digital DLL for CMD/ADDR group 5

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x558 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx00000000000xxxxxxx00000)

Bit	Reset	Description
22:12	0x0	XFORM_ADDR5_OFFS: [PMC]
4:0	0x0	XFORM_ADDR5_MULT: [PMC]

16.7.2.244 EMC_DLI_ADDR_TRIM2_0

Configure Digital DLL

Offset: 0x55c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:20	X	ADDR5_CURRENT_TRIM_VAL
19:10	X	ADDR4_CURRENT_TRIM_VAL
9:0	X	ADDR3_CURRENT_TRIM_VAL

16.7.2.245 EMC_CFG_PIPE_0

Datapipe Configuration Register

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x560 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000xx000000xx00)

Bit	Reset	Description
15	0x0	EMC2PMACRO_CFG_BYPASS_OB_DATAPIPE4: [PMC] 1 = bypass data pipeline stage2 (closer to I/O) between E
14	0x0	EMC2PMACRO_CFG_BYPASS_OB_DATAPIPE3: [PMC] 1 = bypass data pipeline stage2 (closer to I/O) between E
13	0x0	EMC2PMACRO_CFG_BYPASS_OB_DATAPIPE2: [PMC] 1 = bypass data pipeline stage2 (closer to I/O) between E
12	0x0	EMC2PMACRO_CFG_BYPASS_OB_DATAPIPE1: [PMC] 1 = bypass data pipeline stage1 (closer to core) between EMC and data pad-macro.
9	0x0	EMC2PMACRO_CFG_BYPASS_OB_ADDRPIPE2: [PMC] 1 = bypass address pipeline stage between EMC and address pad-macro.
8	0x0	EMC2PMACRO_CFG_BYPASS_OB_ADDRPIPE1: [PMC] 1 = bypass address pipeline stage between EMC and address pad-macro.
7	0x0	EMC2PMACRO_CFG_BYPASS_IB_DATAPIPE4: [PMC] 1 = bypass data pipeline stage2 (closer to I/O) between E
6	0x0	EMC2PMACRO_CFG_BYPASS_IB_DATAPIPE3: [PMC] 1 = bypass data pipeline stage2 (closer to I/O) between E
5	0x0	EMC2PMACRO_CFG_BYPASS_IB_DATAPIPE2: [PMC] 1 = bypass data pipeline stage2 (closer to I/O) between E
4	0x0	EMC2PMACRO_CFG_BYPASS_IB_DATAPIPE1: [PMC] 1 = bypass data pipeline stage1 (closer to core) between EMC and data pad-macro.
1	0x0	EMC2PMACRO_CFG_BYPASS_IB_ADDRPIPE2: [PMC] 1 = bypass address pipeline stage between EMC and address pad-macro.

Bit	Reset	Description
0	0x0	EMC2PMACRO_CFG_BYPASS_IB_ADDRPIPE1: [PMC] 1 = bypass address pipeline stage between EMC and address pad-macro.

16.7.2.246 EMC_QPOP_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

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: 0x564 | Read/Write: R/W | Reset: 0x00000006 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000110)

Bit	Reset	Description
5:0	0x6	QPOP: [PMC] time from read command to pop data from the pad macro FIFO. 45 = MAX

16.7.2.247 EMC_QUSE_WIDTH_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x568 | Read/Write: R/W | Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0011)

Bit	Reset	Description
3:0	0x3	QUSE_DURATION: [PMC] Additional QUSE duration apart from default BL/2

16.7.2.248 EMC_PUTERM_WIDTH_0

DRAM timing parameter

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x56c | Read/Write: R/W | Reset: 0x00000005 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0101)

Bit	Reset	Description
3:0	0x5	PUTERM_DURATION: [PMC] Additional PUTERM duration apart from default BL/2

16.7.2.249 EMC_BGBIAS_CTL0_0

BGBIAS Pad controls

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x570 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	0x0	BIAS0_DSC_E_PWRD_IBIAS_RX: [PMC] Bias current going out to brick receiver
2	0x0	BIAS0_DSC_E_PWRD_IBIAS_VTTGEN: [PMC] Active High. Disables biasing current going out to VTTGEN cells
1	0x0	BIAS0_DSC_E_PWRD: [PMC] Active High. Disables all DC current paths within entire cell.

Bit	Reset	Description
0	0x0	BIAS0_DSC_BG_SEL_N: [PMC] Active low; 0= select BandGap reference current;

16.7.2.250 EMC_PUTERM_ADJ_0

PUTERM controls

This register is shadowed: see usage note at the top of Section 16.7.2

Offset: 0x574 | Read/Write: R/W | Reset: 0x00000081 (0bxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx01)

Bit	Reset	Description
7	ENABLED	CFG_PUTERM_EXTEND_HALF_CLK: [PMC] extend the PUTERM window by half a clock 0 = DISABLED 1 = ENABLED
1:0	0x1	CFG_PUTERM_LATE: [PMC] Half-cycle increments

16.7.2.251 EMC_CA_TRAINING_SP1_RESULT1_0

Offset: 0x5f0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CA_TRIM_RESULT_SP1_31_0: Indicates which trim setting(s) pass the CA-to-DQ comparison.

16.7.2.252 EMC_CA_TRAINING_SP1_RESULT2_0

Offset: 0x5f4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CA_TRIM_RESULT_SP1_63_32: Indicates which trim setting(s) pass the CA-to-DQ comparison.

16.7.2.253 EMC_CA_TRAINING_SP1_RESULT3_0

Offset: 0x5f8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CA_TRIM_RESULT_SP1_95_64: Indicates which trim setting(s) pass the CA-to-DQ comparison.

16.7.2.254 EMC_CA_TRAINING_SP1_RESULT4_0

Offset: 0x5fc | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CA_TRIM_RESULT_SP1_127_96: Indicates which trim setting(s) pass the CA-to-DQ comparison.

[THIS PAGE INTENTIONALLY LEFT BLANK]

17.0 AHB

17.1 AHB Bus

The AHB Bus conforms to the *AMBA Specification (Rev 2.0) Advanced High-performance Bus (AHB)* architecture as published by ARM.

AHB is a 32-bit multi-master bus. Despite what its name implies, it is a second tier bus in the Tegra® K1 processor, slower and less flexible than the AXI bus used by the CPU, or the memory client interface used by the high-speed devices.

The Master clients on the AHB are:

- The AHB memory controller slave, which is the interface between the AHB bus and the Memory Controller
- The AVP crossbar, both as a master and as a slave. This is the path to IRAM from AHB devices, and also the AVP and CPU path to AHB devices. Neither the CPU complex nor the AVP can access DRAM through this path.
- The APB DMA controller, used to provide data transfer between APB devices and memory
- The USB-OTG controller
- MIPI HSI controller
- ARC controller
- The security engine (SE), both as a master and as a slave
- BSEA and BSEV bitstream engines
- The CoreSight™ debug controller
- The deprecated AHB DMA controller master
- SNOR AHB master
- DDS
- USB2 and USB3 controllers

The Slave clients on the AHB are:

- XBAR
- DRAM
- USB-OTG, USB2, USB3
- SNOR
- TZRAM

AHB in the Tegra K1 processor supports secure access to memory, using the same secure bit mechanism used by the CPU TrustZone security mechanism. This is supported by the SE, which can make secure accesses to memory.

17.2 AHB Bus Arbiter

The AHB Arbiter controls AHB bus master arbitration. This effectively forms a second level of arbitration for access to the memory controller through the AHB Slave Memory device, although AHB masters in some circumstances can use other AHB slaves, in particular they can access IRAM.

The controls presented here are largely for diagnostic purposes only. For suspect AHB performance problems, try disabling the other masters for the device with performance issues. In normal operation bus parking is usually enabled, and all the masters are enabled.

Also see the AHB slave interface registers, where the AHB pre-fetch logic can be configured to enhance performance for devices doing sequential read access from DRAM.

Each AHB master is assigned to either the high or low priority bin.

17.2.1 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.

17.2.2 AHB Arbiter Registers

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

In prior generation Tegra processors, this location used to hold a configuration register for COP_CACHE, but COP_CACHE has been replaced with AVP_CACHE, which has a region in a different section. For software compatibility, this location is reserved.

17.2.2.1 AHB_ARBITRATION_DISABLE_0

The AHB arbitration control register allows user to tweak arbitration behavior of the AHB arbiter.

- Enable bus parking. This keeps the last serviced AHB master on the bus granted so that it can start another transaction faster. If bus parking is disabled, no AHB master will be able to start a new transaction until the arbitration is done and the master is granted the bus.
- Allows the user to specifically disable an AHB master from arbitrating on the AHB bus.

AHB Arbitration Controller

Offset: 0x4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00xxxxxxxx0xx000000x0xxx00000000)

Bit	Reset	Description
31	0x0	DIS_BUS_PARK: 1 = Disable bus parking. 0 = ENABLE 1 = DISABLE
30	0x0	DIS_PENULTIMATE_ARB: 1 = Disable arbitration on the second to last transfer.
21	0x0	MIPIHSI: 1 = Disable MIPIHSI from arbitration. 0 = ENABLE 1 = DISABLE
18	0x0	USB2: 1 = Disable USB2 from arbitration. 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
17	0x0	USB3: 1 = Disable USB3 from arbitration. 0 = ENABLE 1 = DISABLE
16	0x0	BSEA: 1 = Disable BSEA from arbitration. 0 = ENABLE 1 = DISABLE
15	0x0	DDS: 1 = Disable DDS from arbitration. 0 = ENABLE 1 = DISABLE
14	0x0	SE: 1 = Disable SE from arbitration. 0 = ENABLE 1 = DISABLE
13	0x0	BSEV: 1 = Disable BSEV from arbitration. 0 = ENABLE 1 = DISABLE
11	0x0	SNOR: 1 = Disable SNOR from arbitration. 0 = ENABLE 1 = DISABLE
7	0x0	APBDMA: 1 = Disable APB-DMA from arbitration. 0 = ENABLE 1 = DISABLE
6	0x0	USB: 1 = Disable USB from arbitration. 0 = ENABLE 1 = DISABLE
5	0x0	AHBDMA: 1 = Disable AHB-DMA from arbitration. 0 = ENABLE 1 = DISABLE
4	0x0	ARC: 1 = Disable ARC from arbitration. 0 = ENABLE 1 = DISABLE
3	0x0	CSITE: 1 = Disable CoreSight from arbitration. 0 = ENABLE 1 = DISABLE
2	0x0	VCP: 1 = Disable VCP from arbitration. 0 = ENABLE 1 = DISABLE
1	0x0	COP: 1 = Disable COP from arbitration. 0 = ENABLE 1 = DISABLE
0	0x0	CPU: 1 = Disable CPU from arbitration. 0 = ENABLE 1 = DISABLE

17.2.2.2 AHB_ARBITRATION_PRIORITY_CTRL_0

The AHB arbiter implements a 2-level priority scheme. In the first level, arbitration is determined between the high and low priority group according to the priority weight; the higher the weight, the higher the winning rate of the high priority group.

In the second level, within each of the high/low priority group, arbitration is determined in a round-robin fashion.

AHB Arbitration Priority Control Register

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000) | Default: 0xe0000000

Bit	Reset	SW Default	Description
31:29	0x0	0x7	AHB_PRIORITY_WEIGHT: AHB priority weight count. This 3-bit field is used to control the amount of attention (weight) given to the high priority group before switching to the low priority group.
28:0	0x0	NONE	AHB_PRIORITY_SELECT: 0 = low priority group.

17.2.2.3 AHB_ARBITRATION_USR_PROTECT_0

USR Protection Register

Offset: 0xc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000x0000)

Bit	Reset	Description
8	0x0	CACHE: Abort on USR mode access to Cache memory space 0 = ABT_DIS 1 = ABT_EN
7	0x0	ROM: Abort on USR mode access to internal ROM memory space 0 = ABT_DIS 1 = ABT_EN
6	0x0	APB: Abort on USR mode access to APB memory space 0 = ABT_DIS 1 = ABT_EN
5	0x0	AHB: Abort on USR mode access to AHB memory space 0 = ABT_DIS 1 = ABT_EN
3	0x0	IRAMD: Abort on USR mode access to iRAMd memory space 0 = ABT_DIS 1 = ABT_EN
2	0x0	IRAMC: Abort on USR mode access to iRAMc memory space 0 = ABT_DIS 1 = ABT_EN
1	0x0	IRAMB: Abort on USR mode access to iRAMb memory space 0 = ABT_DIS 1 = ABT_EN
0	0x0	IRAMA: Abort on USR mode access to iRAMa memory space 0 = ABT_DIS 1 = ABT_EN

17.3 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.

17.3.1 AHB Gizmo Registers

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

17.3.1.1 AHB_GIZMO_AHB_MEM_0

AHB Master/Slave Gizmo Register

Given below is a description of each configuration field, what they are used for and the pros and cons of using them.

> **Note:** These configuration bits are meant to be changed only when the particular gizmo you want to change is idle (not being used). Changing the configuration bits on-the-fly while that gizmo is active can hang the system.

AHB gizmo configuration bit definition:

(1) AHB Master Gizmo

1. MAX_AHB_BURSTSIZE: Controls the maximum burst size that this gizmo can generate on the AHB bus. For the current system, this field should be set to burst-of-8.
2. IMMEDIATE: Controls how quickly an AHB write request on the bus can start.
   - If 1, the gizmo will start an AHB write request once the IP side provides one beat of data of a burst. For IP that can provide quick continuous burst data, this setting provides parallelism between AHB request and IP data transfer. However, for IP that cannot provide quick burst data, this setting will force this gizmo to hold the AHB bus longer, reducing bus efficiency.
   - If 0, gizmo will wait until all beats of data of a burst are provided before starting an AHB write request. With this setting, AHB bus efficiency is realized but IP latency and efficiency may be reduced.
3. RD_DATA: Controls how read data will be returned from the gizmo to the IP side.
   - If 1, all beats of data of a burst have to be available before it indicates to the IP logic that read data is ready. This setting ensures there is no wait-state (bubble) between read data burst, but it will increase latency.
   - If 0, each beat of data of a burst will be sent from gizmo to the IP logic immediately. This setting will reduce latency, but can create non-consecutive data burst or bubble between data.
4. REQ_NEG_CNT: Provides a way to limit (in terms of number of AHB bus clock) how fast/often this master can request the AHB bus. The bigger the number, the slower the rate of AHB request.

(2) AHB Slave Gizmo

1. ENABLE_SPLIT: Controls enabling the split feature of the AHB bus.
   - If 1, the gizmo will always generate split response for a read. If 'DONT_SPLIT_AHB_WR' is set to 0, gizmo also generates split response if it can no longer accept a write request from the AHB master. 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 to get to the data it asked for.

- If 0, the gizmo will not generate split response. Instead, it will hold on to the AHB bus until all the requested read data is returned back to the AHB master. This setting will reduce read latency to the master, but decrease AHB bus utilization.

2. FORCE_TO_AHB_SINGLE: Controls how the gizmo treats the AHB master's burst request.
   - If 1, the gizmo will break up the burst request internally into individual single word requests.
   - If 0, the gizmo will burst request internally as burst request.
3. ENB_FAST_REARBITRATE: Controls when the gizmo can allow the original read requested AHB master to re-arbitrate for the AHB bus again so the AHB master can retrieve the originally requested read data.
   - If 1, once first read data of a burst is in the slave gizmo's FIFO, it will allow the original AHB master to re-arbitrate, thus, allowing arbitration to happen in parallel with subsequent read data of a burst. However, if the data burst has wait-states or bubbles, then this setting will decrease AHB bus utilization because the slave gizmo will hold on to the AHB bus longer.
   - If 0, all read data of a burst must be in the slave gizmo's FIFO before it allows the original AHB master to re-arbitrate to retrieve the read data. This increases read data latency, and also increase AHB bus utilization.
4. IP_WR_REQ_IMMEDIATE: Controls when the gizmo will start write data request to the IP logic.
   - If 1, the gizmo will start write request to the IP logic once it gets one write data of a burst from the AHB side. This setting will create non-consecutive/bubbles in a write data burst.
   - If 0, the gizmo will start write request to the IP logic only when it gets all write data of a burst from the AHB side. This setting will create consecutive data burst (no bubbles or wait-states).
5. MAX_IP_BURSTSIZE: Controls the maximum burst size that this gizmo can generate to the IP logic.
6. ACCEPT_AHB_WR_ALWAYS: Controls how the slave gizmo will treat AHB write request.
   - If 1, the gizmo will always accept a write request without checking whether it's FIFOs can accept the write or not. This setting can reduce bus utilization, but can reduce the rate of AHB retry.
   - If 0, the gizmo will check its FIFOs to make sure they have room before accepting the AHB write request. This setting increase bus utilization, but can create a lot of AHB retry.
7. DONT_SPLIT_AHB_WR: Controls whether to split AHB write request when the slave gizmo determines it cannot accept the write request.
   - If 1 and when ENABLE_SPLIT=1, the gizmo will generate split for write if its FIFOs are not ready to accept the AHB write request or data. This setting can improve AHB bus utilization as there are no continuous AHB master retries on the bus.
   - If 0, the gizmo will generate retry response for write if its FIFOs are not ready to accept the AHB write request. Software should leave this bit at 0 since this feature has not been proven.

AHB Gizmo AHB-DMA/Memory Control Register

Offset: 0x10 | Read/Write: R/W | Reset: 0x00020081 (0b00000000xxxxx010xxxxxx0010xxx001) | Default: 0x00000304

Bit	Reset	SW Default	Description
31:24	0x0	NONE	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	0x0	NONE	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.

Bit	Reset	SW Default	Description
			0 = DISABLE 1 = ENABLE
17:16	0x2	NONE	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
9	0x0	ENABLE	EN_USB_WAIT_COMMIT_ON_1K_STALL: Enabling this register and enabling WR_WAIT_COMMIT_ON_1K causes MC requests to be stalled when the current request from the USB is in a different 4KB block with respect to the previous USB request. The register field name indicates 1K but the actual implementation is 4K. 0 = DISABLE 1 = ENABLE
8	0x0	ENABLE	WR_WAIT_COMMIT_ON_1K: writes to ahbslv will only be issued to the MC when all prior writes in different 4KB pages have reached the point of coherency. This is needed because the MC allows re-ordering of transactions. If this is not set, an entity which polls memory for a descriptor or flag to indicate some other memory has been written, may think something is available in memory when it is not. The register field name indicates 1K but the actual implementation is 4K. 0 = DISABLE 1 = ENABLE
7	0x1	NONE	DONT_SPLIT_AHB_WR: AHB slave gizmo (memory controller) – Do not split AHB write transaction 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	NONE	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo (memory controller) - Accept AHB write request always. 1= Always accept AHB write request without checking whether there is room in the queue to store the write data. Bypass Memory Controller AHB slave gizmo write queue. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. Memory controller AHB slave gizmos write queue is used in this case. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x0	ENABLE	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE

Bit	Reset	SW Default	Description
1	0x0	NONE	FORCE_TO_AHB_SINGLE: AHB slave gizmo (memory controller) - Force all AHB transaction to single data request transaction 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	NONE	ENABLE_SPLIT: AHB slave gizmo (memory controller) - Enable splitting AHB transaction. 0 = DISABLE 1 = ENABLE

17.3.1.2 AHB_GIZMO_APB_DMA_0

AHB Gizmo APB-DMA Control Register

Offset: 0x14 │ Read/Write: R/W │ Reset: 0x00020000 (0b00000000xxxxx010xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo (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

17.3.1.3 AHB_MASTER_SWID_0

AHB Master SWID[0] Register

Writes are secure.

Offset: 0x18 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxx0xx000000x0xxxx0000000)

Bit	Reset	Description
21	0x0	MIPIHSI: SWID[0] for MIPIHSI
18	0x0	USB2:SWID[0] for USB2
17	0x0	USB3: SWID[0] for USB3
16	0x0	BSEA:SWID[0] for BSEA

Bit	Reset	Description
15	0x0	DDS:SWID[0] for DDS
14	0x0	SE:SWID[0] for SE
13	0x0	BSEV:SWID[0] for BSEV
11	0x0	NOR: SWID[0] for NOR
6	0x0	USB1:SWID[0] for USB1
5	0x0	AHBDMA:SWID[0] for AHBDMA
4	0x0	ARC: SWID[0] for ARC
3	0x0	CORESIGHT:SWID[0] for CORESIGHT
2	0x0	VCP:SWID[0] for VCP
1	0x0	COP:SWID[0] for COP
0	0x0	CPU:SWID[0] for CPU

17.3.1.4 AHB_GIZMO_USB_0

AHB Gizmo USB Control Register

Offset: 0x20 │ Read/Write: R/W │ Reset: 0x00020087 (0b00000000xxxxx010xxxxxxxx10xxx111)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – do not split AHB write transaction. 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE

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 is enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x1	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x1	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 0 = DISABLE 1 = ENABLE

17.3.1.5 AHB_GIZMO_AHB_XBAR_BRIDGE_0

AHB Gizmo AHB XBAR Bridge Control Register

Offset: 0x24 │ Read/Write: R/W │ Reset: 0x0000008d (0bxxxxxxxxxxxxxxxxxxxxxxxxx10001101)

Bit	Reset	Description
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – do not split AHB write transaction. 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = Always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK

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
2	0x1	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x0	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 0 = DISABLE 1 = ENABLE

17.3.1.6 AHB_GIZMO_CPU_AHB_BRIDGE_0

AHB Gizmo CPU AHB Bridge Control Register

Offset: 0x28 │ Read/Write: R/W │ Reset: 0x00060000 (0b00000000xxxxx110xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master

Bit	Reset	Description
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

17.3.1.7 AHB_GIZMO_COP_AHB_BRIDGE_0

AHB Gizmo COP AHB Bridge Control Register

Offset: 0x2c │ Read/Write: R/W │ Reset: 0x00060000 (0b00000000xxxxx110xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master
18	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

17.3.1.8 AHB_GIZMO_XBAR_APB_CTLR_0

AHB Gizmo XBAR APB Control Register

Offset: 0x30 │ Read/Write: R/W │ Reset: 0x00000008 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx001xxx)

Bit	Reset	Description
5:4	0x0	MAX_IP_BURSTSIZE: AHB slave gizmo - Maximum allowed IP burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
3	0x1	IMMEDIATE: AHB slave gizmo - Start write request to device immediately. 1 = Start write request on the device side as soon as the AHB master puts data into the gizmos queue. 0 = Start the device write request only when the AHB master has placed all write data into the gizmos queue. 0 = DISABLE 1 = ENABLE

17.3.1.9 AHB_GIZMO_VCP_AHB_BRIDGE_0

AHB Gizmo VCP AHB Bridge Control Register

Offset: 0x34 │ Read/Write: R/W │ Reset: 0x00060000 (0b00000000xxxxx110xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	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

17.3.1.10 AHB_MASTER_SWID_1

AHB Master SWID[1] Register

Writes are secure.

Offset: 0x38 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx0xx000000x0xxxx0000000)

Bit	Reset	Description
20	0x0	MIPIHSI: SWID[1] for MIPIHSI
18	0x0	USB2:SWID[1] for USB2
17	0x0	USB3: SWID[1] for USB3
16	0x0	BSEA:SWID[1] for BSEA
15	0x0	DDS:SWID[1] for DDS
14	0x0	SE:SWID[1] for SE
13	0x0	BSEV:SWID[1] for BSEV
11	0x0	NOR:SWID[1] for NOR
6	0x0	USB1:SWID[1] for USB1
5	0x0	AHBDMA:SWID[1] for AHBDMA
4	0x0	ARC: SWID[1] for ARC
3	0x0	CORESIGHT:SWID[1] for CORESIGHT
2	0x0	VCP:SWID[1] for VCP
1	0x0	COP:SWID[1] for COP
0	0x0	CPU:SWID[1] for CPU

17.3.1.11 AHB_GIZMO_SE_0

AHB Gizmo SE Control Register

Offset: 0x50 | Read/Write: R/W | Reset: 0x00020000 (0b00000000xxxxx010xxxxxxxxxxxxxxxx) | Default: 0x00000000

Bit	Reset	SW Default	Description
31:24	0x0	NONE	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	DISABLE	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	NONE	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

Bit	Reset	SW Default	Description
			0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

17.3.1.12 AHB_GIZMO_TZRAM_0

AHB Gizmo AHB TZRAM Control Register

Offset: 0x54 │ Read/Write: R/W │ Reset: 0x00000081 (0bxxxxxxxxxxxxxxxxxxxxxxxx10xxx001)

Bit	Reset	Description
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – Do not split AHB write transaction. 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = Always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x0	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x0	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 0 = DISABLE 1 = ENABLE

17.3.1.13 AHB_GIZMO_BSEV_0

This is the VDE's BSEV master gizmo configuration.

AHB Gizmo BSE Control Register

Offset: 0x64 │ Read/Write: R/W │ Reset: 0x00020000 (0b00000000xxx0x010xxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
20	0x0	RESERVED_GIZMO_BSEV: Reserved for future use

Bit	Reset	Description
18	0x0	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 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

17.3.1.14 AHB_GIZMO_BSEA_0

VDE's BSEA master gizmo configuration

AHB Gizmo SCE Control Register

Offset: 0x74 | Read/Write: R/W | Reset: 0x00020000 (0b00000000xxxxx010xxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device puts data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. NOTE: THIS SHOULD NEVER BE SET TO ENABLE (BSEV requires this bit to be 0) 0 = DISABLE 1 = ENABLE
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

17.3.1.15 AHB_GIZMO_NOR_0

AHB Gizmo AHB NOR Flash Control Register

Offset: 0x78 | Read/Write: R/W | Reset: 0x00020081 (0b00000000xxxxx010xxxxxxx10xxx001)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This 8-bit counter is used to indicate the minimum number of clock counts between requests from this AHB master.
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
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – Do not split AHB write transaction. 1 = Do not ever split AHB write transaction. 0 (and enable_split=1) = Allow AHB write transactions to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo – Always accept AHB write requests. 1 = Always accept AHB write requests without checking whether there is room in the queue to store the write data. 0 = Accept AHB write requests only when there is enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x0	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmo's queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmo's queue. 0 = DISABLE 1 = ENABLE
1	0x0	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transactions to single data request transaction. 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 0 = DISABLE 1 = ENABLE

17.3.1.16 AHB_GIZMO_USB2_0

USB2 master gizmo configuration.

AHB Gizmo USB2 Control Register

Offset: 0x7c │ Read/Write: R/W │ Reset: 0x00020087 (0b00000000xxxxx010xxxxxxxx10xxx111)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device puts data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – Do not split AHB write transaction. 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = Always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x1	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x1	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 0 = DISABLE 1 = ENABLE

17.3.1.17 AHB_GIZMO_USB3_0

USB3 master gizmo configuration.

AHB Gizmo USB3 Control Register

Offset: 0x80 | Read/Write: R/W | Reset: 0x00020087 (0b00000000xxxxx010xxxxxxxx10xxx111)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device puts data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8. 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – Do not split AHB write transaction. 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = Always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x1	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x1	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 0 = DISABLE 1 = ENABLE

17.3.1.18 AHB_GIZMO_DDS_0

AHB Gizmo DDS Control Register

Offset: 0x90 | Read/Write: R/W | Reset: 0x00020000 (0b00000000xxxxx010xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	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

17.3.1.19 AHB_GIZMO_MIPIHSI_0

AHB Gizmo MIPIHSI Control Register

Offset: 0x94 | Read/Write: R/W | Reset: 0x00020000 (0b00000000xxxxx010xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	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

17.3.1.20 AHB_GIZMO_ARC_0

AHB Gizmo ARC Control Register

MC Master Gizmo Configuration

Offset: 0x98 │ Read/Write: R/W │ Reset: 0x00060000 (0b00000000xxxxx110xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	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

17.3.1.21 AHB_AHB_WRQ_EMPTY_0

Offset: 0xc4 │ Read/Write: RO │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	COP_AHB_WRQ_EMPTY
0	X	CPU_AHB_WRQ_EMPTY

17.4 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.

17.4.1 AHB Memory Pre-Fetcher

The AHB memory controller slave contains a pre-fetcher block. This is intended to improve performance for AHB masters performing sequential reads from DRAM. Performance can be a problem because the AHB bus protocol only allows a single outstanding read request from a master, so the round trip latency limits bandwidth. As there are two level of arbitration, firstly on AHB and secondly within the memory controller, this latency can become large on a busy system.

There are eight such pre-fetchers, allowing this function to be active for up to eight AHB masters.

When the pre-fetcher is enabled, the first read request initiates a speculative read of up to 128 bytes, and subsequent linear reads will return the data directly to the AHB master without going through the memory controller to DRAM.

17.4.1.1 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 AHB 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.

17.4.1.2 Pre-Fetch Buffer Coherency

As the pre-fetch buffer contains a copy of data in DRAM, there is an inherent coherency risk if the DRAM data is updated. The hardware invalidation mechanisms provide some protection here, but there remains risk of subtle problems arising from this hardware. If an AHB master shows errors that appear to arise from stale data then a reasonable experiment is to disable the pre-fetcher to see if that cures the problem. If so, the following guidelines may help.

Problems have been seen for control structures such as USB Transfer Descriptors.

Two approaches can resolve this problem:

1. Pad the data

As the pre-fetcher invalidates the buffer for any non-sequential read, placing some padding will prevent pre-fetch issues. A problem can be triggered by using something like this:

```
struct dtd { unsigned HW_descriptor[K] ; } ; struct dtd dtd_array[N] ;
```

This makes the hardware descriptors structures as read by the USB DMA sequential, and so pre-fetchable.
If the definition is modified as:

```
struct dtd { unsigned HW_descriptor[K] ; unsigned Padding ; } ; struct dtd dtd_array[N] ;
```

Then reads by USB DMA of consecutive hardware descriptors become **not** sequential and so there is no coherency issue as the pre-fetch buffer will be invalidated.

2. Invalidate the pre-fetch buffer

After updating a memory structure, the pre-fetch buffer can be invalidated by disabling the pre-fetcher and then immediately re-enabling it.

17.4.2 AHB Memory Controller Slave Registers

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

17.4.2.1 AHB_AHB_MEM_PREFETCH_CFG5_0

0x6000_C0CC: ahbslv prefetch cfg5 --> reset value = 0x1883.0800.

Offset: 0xcc │ Read/Write: R/W │ Reset: 0x14800800 (0b00010100100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable, 0=disable
30:26	0x5	AHB_MST_ID: 0 = CPU

Bit	Reset	Description
		1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = SDIO1 10 = NAND_FLASH 11 = SNOR 12 = HSMMC 13 = BSEV 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = SDIO2 20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 16 will use n=16.
15:0	0x800	INACTIVITY_TIMEOUT

17.4.2.2 AHB_AHB_MEM_PREFETCH_CFG6_0

0x6000_C0D0: ahbslv prefetch cfg6 --> reset value = 0x1883.0800.

Offset: 0xd0 │ Read/Write: R/W │ Reset: 0x18800800 (0b00011000100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable, 0=disable
30:26	0x6	AHB_MST_ID: USB 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = SDIO1 10 = NAND_FLASH 11 = SNOR 12 = HSMMC 13 = BSEV 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = SDIO2

Bit	Reset	Description
		20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 16 will use n=16.
15:0	0x800	INACTIVITY_TIMEOUT

17.4.2.3 AHB_AHB_MEM_PREFETCH_CFG7_0

0x6000_C0D4: ahbslv prefetch cfg7 --> reset value = 0x1883.0800.

Offset: 0xd4 │ Read/Write: R/W │ Reset: 0x1c800800 (0b00011100100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable, 0=disable
30:26	0x7	AHB_MST_ID: USB 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = SDIO1 10 = NAND_FLASH 11 = SNOR 12 = HSMMC 13 = BSEV 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = SDIO2 20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 16 will use n=16.
15:0	0x800	INACTIVITY_TIMEOUT

17.4.2.4 AHB_AHB_MEM_PREFETCH_CFG8_0

0x6000_C0D8: ahbslv prefetch cfg8 --> reset value = 0x1883.0800.

Offset: 0xd8 │ Read/Write: R/W │ Reset: 0x20800800 (0b00100000100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable, 0=disable
30:26	0x8	AHB_MST_ID: USB 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = SDIO1 10 = NAND_FLASH 11 = SNOR 12 = HSMMC 13 = BSEV 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = SDIO2 20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 16 will use n=16.
15:0	0x800	INACTIVITY_TIMEOUT

17.4.2.5 AHB_AHB_MEM_PREFETCH_CFG_X_0

If DISABLE_CHECK_SIZE is 0, then only read requests that have the exact same size as the original read request that kick-started the prefetch process will cause a "hit". In addition, the address must be the exact next one in the sequence.

For instance, if the first request on a prefetch-enabled AHB Master arrives with SIZE=ONE_BYTE and ADDR[31:0]=0x3, then the next access must have SIZE=ONE_BYTE and ADDR[31:0]=0x4 in order to be considered a hit.

If DISABLE_CHECK_SIZE is 1, then a read request will hit as long as the incoming ADDR[31:4]matches the expected_ADDR[31:4]. expected_ADDR[31:4] is always either "last ADDR[31:4]" or "last ADDR[31:4] + 1", where last ADDR is the prior read request actually issued by the AHB Master(as opposed to the last request to the CIF, which could have been a speculative read).

expected_ADDR[31:4] is always "last ADDR[31:4]" unless "SIZE" field in the last request uses the last byte in the 16-byte CIF word.

Offset: 0xdc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15	0x0	DISABLE_ADDR_BNDY_CHK_MST8:
14	0x0	DISABLE_ADDR_BNDY_CHK_MST7:
13	0x0	DISABLE_ADDR_BNDY_CHK_MST6:
12	0x0	DISABLE_ADDR_BNDY_CHK_MST5:
11	0x0	DISABLE_ADDR_BNDY_CHK_MST4:
10	0x0	DISABLE_ADDR_BNDY_CHK_MST3:
9	0x0	DISABLE_ADDR_BNDY_CHK_MST2:
8	0x0	DISABLE_ADDR_BNDY_CHK_MST1
7	0x0	DISABLE_CHECK_SIZE_MASTER8:
6	0x0	DISABLE_CHECK_SIZE_MASTER7:
5	0x0	DISABLE_CHECK_SIZE_MASTER6:
4	0x0	DISABLE_CHECK_SIZE_MASTER5:
3	0x0	DISABLE_CHECK_SIZE_MASTER4:
2	0x0	DISABLE_CHECK_SIZE_MASTER3:
1	0x0	DISABLE_CHECK_SIZE_MASTER2:
0	0x0	DISABLE_CHECK_SIZE_MASTER1:

17.4.2.6 AHB_ARBITRATION_XBAR_CTRL_0

XBAR Control Register

Offset: 0xe0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxx00xxxxxxxxxxxxxx00)

Bit	Reset	Description
16	0x0	MEM_INIT_DONE: Software should set this bit when memory has been initialized 0 = NOT_DONE 1 = DONE
1	0x0	HOLD_DIS: By default CPU accesses to IRAMs will be held if there are any pending requests from the AHB to the IRAMs. This is done to avoid data coherency issues. If software handles coherency then this can be turned off to improve performance. Software writes to modify. 0 = ENABLE 1 = DISABLE
0	0x0	POST_DIS: Software writes to modify 0 = ENABLE 1 = DISABLE

17.4.2.7 AHB_AHB_MEM_PREFETCH_CFG3_0

See the description of the pre-fetcher above. 6000:C0E4: ahbslv prefetch cfg3 --> reset value = 0x1483.0800.

Offset: 0xe4 │ Read/Write: R/W │ Reset: 0x0c800800 (0b00001100100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable. 0=disable
30:26	0x3	AHB_MST_ID: AHBDMA master 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = UNUSED_09 10 = UNUSED_0A 11 = SNOR 12 = HSMMC 13 = BSEV 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = UNUSED_13 20 = UNISED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 26 will use n=26.
15:0	0x800	INACTIVITY_TIMEOUT: 2048 cycles

17.4.2.8 AHB_AHB_MEM_PREFETCH_CFG4_0

See the description of the pre-fetcher above. 6000:C0E8: ahbslv prefetch cfg4 --> reset value = 0x1483.0800.

Offset: 0xe8 │ Read/Write: R/W │ Reset: 0x10800800 (0b00010000100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable. 0=disable
30:26	0x4	AHB_MST_ID: AHBDMA master 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08

Bit	Reset	Description
		9 = UNUSED_09 10 = UNUSED_0A 11 = SNOR 12 = HSMMC 13 = BSEV 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = UNUSED_13 20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 26 will use n=26.
15:0	0x800	INACTIVITY_TIMEOUT: 2048 cycles

17.4.2.9 AHB_AVP_PPCS_RD_COH_STATUS_0

0x6000_C0EC: ARM7 outstanding rd/wr

Offset: 0xec │ Read/Write: RO │ Reset: 0x000X000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
16	X	RDS_OUTSTANDING
0	X	WRS_OUTSTANDING

17.4.2.10 AHB_AHB_MEM_PREFETCH_CFG1_0

NV_ahbslvmem prefetch

The NV_ahbslvmem prefetch feature reduces latency and improves overall bandwidth for AHB Masters doing reads to SDRAM. AHB only allows one outstanding read transaction at a time. When enabled, and kick-started by a first read request (which will "miss" since there would be nothing in the prefetch FIFO) the prefetcher makes speculative read requests to the memory controller in consecutive progression of linear address. Without this feature, an AHB Master will suffer roundtrip latency through the gizmo, PPCS, CIF, MC arbitration, and all the read data passing for every AHB transaction.

Address Boundaries for prefetching will stop the sequential prefetch process from making additional speculations. This is useful to help prevent coherency issues -- software needs ways to stop the prefetcher from prefetching data before the data has been written in memory. If an AHB master can be known to make 256 byte transfers that are aligned accesses, ADDR_BNDRY should be set to a value of 4.

INACTIVITY_TIMEOUT is intended to prevent coherency problems. If no read request from the prefetch-enabled master is observed after the number of cycles specified in the inactivity timeout counter, then any speculatively prefetched read data is invalidated.

There are eight AHB masters that can be enabled for prefetching. Each prefetch buffer can hold up to 8 entries (of 128 bits each entry) of prefetched data.

0x6000_C0F0: ahbslv prefetch cfg1 --> reset value = 0x1483.0800.

Offset: 0xf0 │ Read/Write: R/W │ Reset: 0x04800800 (0b00000100100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable, 0=disable
30:26	0x1	AHB_MST_ID: AHBDMA master 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = UNUSED_09 10 = UNUSED_0A 11 = SNOR 12 = HSMMC 13 = BSEV 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = UNUSED_13 20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 26 will use n=26.
15:0	0x800	INACTIVITY_TIMEOUT: 2048 cycles

17.4.2.11 AHB_AHB_MEM_PREFETCH_CFG2_0

See the description of the pre-fetcher above. 0x6000_C0F4: ahbslv prefetch cfg2 --> reset value = 0x1883.0800.

Offset: 0xf4 │ Read/Write: R/W │ Reset: 0x08800800 (0b00001000100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable, 0=disable

Bit	Reset	Description
30:26	0x2	AHB_MST_ID: AHBDMA master 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = UNUSED_09 10 = UNUSED_0A 11 = SNOR 12 = HSMMC 13 = BSEV 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = UNUSED_13 20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 26 will use n=26.
15:0	0x800	INACTIVITY_TIMEOUT

17.4.2.12 AHB_AHBSLVMEM_STATUS_0

0x6000_C0F8: ahbslv outstanding rd, rdque_empty status

Offset: 0xf8 │ Read/Write: RO │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	PPCS_RDS_OUTSTANDING
0	X	GIZMO_IP_RDQUE_EMPTY

17.4.2.13 AHB_ARBITRATION_AHB_MEM_WRQUE_MST_ID_0

0x6000_C0FC: AHB Memory Write Queue AHB Master ID Register

Offset: 0xfc │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	AHB_MASTER_ID: 0 = There is no write data in the write queue from that AHB master.

17.4.2.14 AHB_ARBITRATION_CPU_ABORT_ADDR_0

0x6000_C100: CPU Abort Address Register

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

Bit	Reset	Description
31:0	X	ADDR: Instruction Address which caused the abort.

17.4.2.15 AHB_ARBITRATION_CPU_ABORT_INFO_0

0x6000_C104: CPU Abort Info Register

Offset: 0x104 │ Read/Write: RO │ Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15	X	IRAMA: Abort occurred due to an iRAMa protection violation 0 = ABT_DIS 1 = ABT_EN
14	X	IRAMB: Abort occurred due to an iRAMb protection violation 0 = ABT_DIS 1 = ABT_EN
13	X	IRAMC: Abort occurred due to an iRAMc protection violation 0 = ABT_DIS 1 = ABT_EN
12	X	IRAMD: Abort occurred due to an iRAMd protection violation 0 = ABT_DIS 1 = ABT_EN
11	X	INV_IRAM: Abort occurred due to an access to invalid iRAM address space 0 = ABT_DIS 1 = ABT_EN
10	X	PPSB: Abort occurred due to a PPSB protection violation 0 = ABT_DIS 1 = ABT_EN
9	X	APB: Abort occurred due to an APB protection violation 0 = ABT_DIS 1 = ABT_EN
8	X	AHB: Abort occurred due to an AHB protection violation 0 = ABT_DIS 1 = ABT_EN
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=halfword 10=word 0 = BYTE_ABT 1 = HWORD_ABT 2 = WORD_ABT

17.4.2.16 AHB_ARBITRATION_COP_ABORT_ADDR_0

0x6000_C108: CPU Abort Address Register

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

Bit	Reset	Description
31:0	X	ADDR: Instruction Address which caused the abort

17.4.2.17 AHB_ARBITRATION_COP_ABORT_INFO_0

0x6000_C10C: COP Abort Info Register

Offset: 0x10c │ Read/Write: RO │ Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15	X	IRAMA: Abort occurred due to an iRAMa protection violation 0 = ABT_DIS 1 = ABT_EN
14	X	IRAMB: Abort occurred due to an iRAMb protection violation 0 = ABT_DIS 1 = ABT_EN
13	X	IRAMC: Abort occurred due to an iRAMc protection violation 0 = ABT_DIS 1 = ABT_EN

Bit	Reset	Description
10	X	PPSB: Abort occurred due to a PPSB protection violation 0 = ABT_DIS 1 = ABT_EN
9	X	APB: Abort occurred due to an APB protection violation 0 = ABT_DIS 1 = ABT_EN
8	X	AHB: Abort occurred due to an AHB protection violation 0 = ABT_DIS 1 = ABT_EN
7	X	CACHE: Abort occurred due to a Cache protection violation 0 = ABT_DIS 1 = ABT_EN
6	X	PROTECTION: TRUE for any protection violation 0 = ABT_DIS 1 = ABT_EN
5	X	ALIGN: TRUE for abort caused by Misalignment (i.e., word access at odd byte address) 0 = ABT_DIS 1 = ABT_EN
4	X	BADSIZE: TRUE for abort caused by Bad Size (i.e., word access at odd byte address) 0 = ABT_DIS 1 = ABT_EN
3	X	WRITE: Aborted transaction was a Write 0 = ABT_DIS 1 = ABT_EN
2	X	DATA: Aborted transaction was a Data access 0 = ABT_DIS 1 = ABT_EN
1:0	X	SIZE: Aborted transaction Request Size 00=byte 01=halfword 10=word 0 = BYTE_ABT 1 = HWORD_ABT 2 = WORD_ABT

17.4.2.18 AHB_AHB_SPARE_REG_0

0x6000_C110: AHB Spare Register Bits

Offset: 0x110 │ Read/Write: R/W │ Reset: 0x00000060 (0b00000000000000000000xxxxx1100000)

Bit	Reset	Description
31:12	0x0	AHB_SPARE_REG: AHB Spare Register Note: For Tegra K1 64-bit processors, bit 31 is used for Microcode-Boot ROM handshaking. See the Boot Process chapter in the Tegra K1 64-Bit Processor TRM for more information.
4:0	0x0	CSITE_PADMACRO_TRIM_SEL: Trimmer select register for CoreSight clock pad macro.

17.4.2.19 AHB_XBAR_SPARE_REG_0

0x6000_C114: XBAR Spare Register Bits

Offset: 0x114 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:11	0x0	UNUSED: XBAR Diagnostic Register Spare Bits
10	0x0	DISABLE_COP_BYTE_WR: XBAR Diagnostic Register to disable COP byte writes
9	0x0	DISABLE_XUSB_DEV_BYTE_WR: XBAR Diagnostic Register to disable XUSB dev byte writes
8	0x0	DISABLE_XUSB_HOST_BYTE_WR: XBAR Diagnostic Register to disable XUSB host byte writes
7	0x0	DISABLE_SDMMC_BYTE_WR: XBAR Diagnostic Register to disable SDMMC byte writes
6	0x0	DISABLE_XBAR_APB_BYTE_WR: XBAR Diagnostic Register to disable byte writes to APB slaves
5	0x0	MASK_PEND_IRAM_REQ: XBAR Diagnostic Register to mask pending IRAM request while arbitration
4	0x0	KILL_NEXT_REQ_ON_ABORT_UCQ: XBAR Diagnostic Register to kill next request on ABORT from UCQ
3	0x0	KILL_NEXT_REQ_ON_ABORT_AHB: XBAR Diagnostic Register to kill next request on ABORT from AHB Bridge
2	0x0	KILL_NEXT_REQ_ON_ABORT_VCP2: XBAR Diagnostic Register to kill next request on ABORT from VCP2
1	0x0	KILL_NEXT_REQ_ON_ABORT_COP: XBAR Diagnostic Register to kill next request on ABORT from COP
0	0x0	KILL_NEXT_REQ_ON_ABORT_APC: XBAR Diagnostic Register to kill next request on ABORT from APC

17.4.2.20 AHB_AVPC_MCCIF_FIFOCTRL_0

Note: The FIFO timing aspects of this register are no longer supported, but are retained for software compatibility.

The clock override/ovr_mode fields of this register control the second-level clock gating for the client and MC sides of the MCCIF. All clock gating is enabled by default.

A '1' written to the rclk/wclk override field will result in one of the following:

- With wclk/rclk override mode = LEGACY, the clock reverts to legacy mode of operation where the clock is on whenever the client clock is enabled.
- With wclk/rclk override mode = ON, the clock is always on inside the MCCIF and PC

A '1' written to the CCLK override field keeps the client's clock always on inside the MCCIF.

Memory Client Interface FIFO Control (where applicable) and Clock Gating Control Register

Offset: 0x120 | Byte Offset: 0x480 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000xxxxxxxxxxxx0000)

Bit	Reset	Description
20	LEGACY	AVPC_RCLK_OVR_MODE: 0 = LEGACY 1 = ON
19	LEGACY	AVPC_WCLK_OVR_MODE: 0 = LEGACY 1 = ON
18	0x0	AVPC_CCLK_OVERRIDE
17	0x0	AVPC_RCLK_OVERRIDE
16	0x0	AVPC_WCLK_OVERRIDE
3	DISABLE	AVPC_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	AVPC_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	AVPC_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	AVPC_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

17.4.2.21 AHB_TIMEOUT_WCOAL_AVPC_0

Note: Write coalescing is no longer supported by the MCCIF clients. Registers are retained for software compatibility but are not used by the hardware.

Write Coalescing Time-Out Register

Offset: 0x124 | Byte Offset: 0x490 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00110010)

Bit	Reset	Description
7:0	0x32	AVPCARM7W_WCOAL_TMVAL

17.4.2.22 AHB_MPCORELP_MCCIF_FIFOCTRL_0

Note: The FIFO timing aspects of this register are no longer supported, but are retained for software compatibility

The clock override/ovr_mode fields of this register control the second-level clock gating for the client and MC sides of the MCCIF. All clock gating is enabled by default.

A '1' written to the rclk/wclk override field will result in one of the following:

- With wclk/rclk override mode = LEGACY, the clock reverts to legacy mode of operation, where the clock is on whenever the client clock is enabled.

- With wclk/rclk override mode = ON, the clock is always on inside MCCIF and PC.

A '1' written to the cclk override field keeps the client's clock always on inside the MCCIF.

Memory Client Interface FIFO Control (where applicable) and Clock Gating Control Register

Offset: 0x128 | Byte Offset: 0x4a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx00000xxxxxxxxxxxxxxxx)

Bit	Reset	Description
20	LEGACY	SYS_REGS_MPCORELP_RCLK_OVR_MODE: 0 = LEGACY 1 = ON
19	LEGACY	SYS_REGS_MPCORELP_WCLK_OVR_MODE: 0 = LEGACY 1 = ON
18	0x0	SYS_REGS_MPCORELP_CCLK_OVERRIDE
17	0x0	SYS_REGS_MPCORELP_RCLK_OVERRIDE
16	0x0	SYS_REGS_MPCORELP_WCLK_OVERRIDE

17.4.2.23 AHB_MPCORE_MCCIF_FIFOCTRL_0

Note: The FIFO timing aspects of this register are no longer supported, but are retained for software compatibility

The clock override/ovr_mode fields of this register control the second-level clock gating for the client and MC sides of the MCCIF. All clock gating is enabled by default.

A '1' written to the rclk/wclk override field will result in one of the following:

- With wclk/rclk override mode = LEGACY, the clock reverts to legacy mode of operation, where the clock is on whenever the client clock is enabled.
- With wclk/rclk override mode = ON, the clock is always on inside MCCIF and PC.

A '1' written to the cclk override field keeps the client's clock always on inside the MCCIF.

Memory Client Interface FIFO Control (where applicable) and Clock Gating Control Register

Offset: 0x12c | Byte Offset: 0x4b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx00000xxxxxxxxxxxxxxxx)

Bit	Reset	Description
20	LEGACY	SYS_REGS_MPCORE_RCLK_OVR_MODE: 0 = LEGACY 1 = ON
19	LEGACY	SYS_REGS_MPCORE_WCLK_OVR_MODE: 0 = LEGACY 1 = ON
18	0x0	SYS_REGS_MPCORE_CCLK_OVERRIDE
17	0x0	SYS_REGS_MPCORE_RCLK_OVERRIDE
16	0x0	SYS_REGS_MPCORE_WCLK_OVERRIDE

18.0 APB

18.1 APB Miscellaneous Registers

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

This section describes a number of system control registers that are grouped together in the aptly named miscellaneous registers section. These registers are all present on the APB bus.

18.1.1 JTAG Configuration Register

18.1.1.1 APB_MISC_PP_CONFIG_CTL_0

Configuration Control Register

The use of the SO bits below has been deprecated. Keep these bits disabled at all times.

Offset: 0x24 │ Read/Write: R/W │ Reset: 0x00000040 (0bxxxxxxxxxxxxxxxxxxxxxxxx01xxxx00)

Bit	Reset	Description
7	DISABLE	TBE: 0 = Disable ; 1 = Enable RTCK Daisy chaining 0 = DISABLE 1 = ENABLE
6	ENABLE	JTAG: 0 = Disable Debug ; 1 = Enable JTAG DBGEN 0 = DISABLE 1 = ENABLE
1	DISABLE	XBAR_SO_DEFAULT: Deprecated -- keep disabled -- default SO bit for non-CPU XBAR clients 0 = DISABLE 1 = ENABLE
0	DISABLE	CPU_XBAR_SO_ENABLE: Deprecated -- keep disabled --Enable CPU SO bit to propagate to XBAR 0 = DISABLE 1 = ENABLE

18.1.1.2 APB_MISC_PP_PINMUX_GLOBAL_0_0

Global Pinmux Control Register

When both CLAMP_INPUTS_WHEN_TRISTATED is set to ENABLE and the TRISTATE field is set to TRISTATE (1b1), the inputs to the core are clamped to zero. Software must set this bit before taking any controller using a pinmuxed pin out of reset.

Offset: 0x40 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	DISABLE	CLAMP_INPUTS_WHEN_TRISTATED: 0 = Do not clamp inputs when tristated; 1 = Clamp inputs when tristated 0 = DISABLE 1 = ENABLE

18.1.2 PULL_UP/PULL_DOWN Control Register

Controls the pull_up/pull_down inputs of the pads. Used to eliminate external components on interfaces that need pullup/pulldown. Those are weak internal pullup/pulldowns

- NORMAL setting means pad is neither pulled up (driving weak 1) or pulled down (driving weak 0)
- PULL_DOWN selection means pad is driving weak 0
- PULL_UP selection means pad is driving weak 1

> **WARNING!** Driving an internal pull-up when the pad has an external pull-down and vice versa will cause a significant increase in power consumption.

Table 59: 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

18.1.2.1 APB_MISC_PP_PULLUPDOWN_REG_C_0

Offset: 0xa8 | Read/Write: R/W | Reset: 0x00000000 (0b000000xxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	NORMAL	XM2C_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
29:28	NORMAL	XM2D_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
27:26	NORMAL	DDRC_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

18.1.2.2 APB_MISC_SC1X_PADS_VIP_VCLKCTRL_0

VCLK Control Register

Enable VCLK pad to get external clock for VI.

Offset: 0x428 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	INVERSION: VCLK invert enable 0 = DISABLE 1 = ENABLE
0	0x0	IE: VCLK input enable 0 = DISABLE 1 = ENABLE

18.1.2.3 APB_MISC_GP_HIDREV_0

Chip ID Revision Register

Offset: 0x804 │ Read/Write: RO │ Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	MINORREV: Chip ID minor revision
15:8	X	CHIPID: Chip ID
7:4	X	MAJORREV: Chip ID major revision (0: Emulation, 1-15: Silicon) 0 = EMULATION 1 = A01
3:0	X	HIDFAM: Chip ID family register. 0 = GPU 1 = HANDHELD 2 = BR_CHIPS 3 = CRUSH 4 = MCP 5 = CK 6 = VAIO 7 = HANDHELD_SOC

18.1.3 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 60: Low Power Mode Functions

LPMD1	LPMD0	Power Mode Selected
0	0	X/8 Current Setting. Lowest Power or Current (8*Z ohm)
0	1	X/4 Current Setting (4*Z ohm)
1	0	X/2 Current Setting (2*Z ohm)
1	1	X Current Setting (Z ohm = 50 ohm). Highest Power or Current.

- CAL_DRVDN - drive down (falling edge) - Driver Output Pull-Down drive strength code.
- CAL_DRVUP - drive up (rising edge) - Driver Output Pull-Up drive strength code. Works with combination of LMPD bits. For lower power modes, higher drive strength are masked. See table below:

LPMD1	LPMD0	CAL_DRV*4	CAL_DRV*3	CAL_DRV*2	CAL_DRV*1	CAL_DRV*0
0	0	Masked to 0	Masked to 0	Masked to 0	Pass Code	Pass Code
0	1	Masked to 0	Masked to 0	Pass Code	Pass Code	Pass Code
1	0	Masked to 0	Pass Code	Pass Code	Pass Code	Pass Code
1	1	Pass Code	Pass Code	Pass Code	Pass Code	Pass Code

- DRVDN_SLWR - Driver Output Rising Edge Slew 2-bit control code.
  Code 11 is least slewing of signal, code 00 is highest slewing of the signal.
- DRVUP_SLWF -Driver Output Falling Edge Slew 2-bit control code.
  Code 11 is least slewing of signal, code 00 is highest slewing of the signal.

18.1.3.1 APB_MISC_GP_MIPI_PAD_CTRL_0

Offset: 0x820 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0x)

Bit	Reset	Description
1	CSI	DSIB_MODE: 0 = CSI 1 = DSI

18.1.3.2 APB_MISC_GP_AOCFG1PADCTRL_0

AOCFG1 Pad Control Register

Offset: 0x868 │ Read/Write: R/W │ Reset: 0xf0e09000 (0b1111xxx01110xxx01001xxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_AOCFG1_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_AOCFG1_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_AOCFG1_CAL_DRVUP
16:12	0x9	CFG2TMC_AOCFG1_CAL_DRVDN
3	0x0	CFG2TMC_AOCFG1_SCHMT_EN: AOCFG1 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_AOCFG1_HSM_EN: AOCFG1 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.3 APB_MISC_GP_AOCFG2PADCTRL_0

AOCFG2 Pad Control Register

Offset: 0x86c │ Read/Write: R/W │ Reset: 0xf0e09000 (0b1111xxx01110xxx01001xxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_AOCFG2_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_AOCFG2_CAL_DRVDN_SLWR

Bit	Reset	Description
24:20	0xe	CFG2TMC_AOCFG2_CAL_DRVUP
16:12	0x9	CFG2TMC_AOCFG2_CAL_DRVDN
3	0x0	CFG2TMC_AOCFG2_SCHMT_EN: AOCFG2 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_AOCFG2_HSM_EN: AOCFG2 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.4 APB_MISC_GP_ATCFG1PADCTRL_0

ATCFG1 Pad Control Register

Offset: 0x870 │ Read/Write: R/W │ Reset: 0x01318000 (0b0000x0010011x0011000xxxx00xx00xx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_ATCFG1_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_ATCFG1_CAL_DRVDN_SLWR
26:20	0x13	CFG2TMC_ATCFG1_CAL_DRVUP
18:12	0x18	CFG2TMC_ATCFG1_CAL_DRVDN
7:6	0x0	CFG2TMC_ATCFG1_DRV_TYPE
3	0x0	CFG2TMC_ATCFG1_SCHMT_EN: ATCFG1 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_ATCFG1_HSM_EN: ATCFG1 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.5 APB_MISC_GP_ATCFG2PADCTRL_0

ATCFG2 Pad Control Register

Offset: 0x874 │ Read/Write: R/W │ Reset: 0x01318000 (0b0000x0010011x0011000xxxx00xx00xx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_ATCFG2_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_ATCFG2_CAL_DRVDN_SLWR
26:20	0x13	CFG2TMC_ATCFG2_CAL_DRVUP
18:12	0x18	CFG2TMC_ATCFG2_CAL_DRVDN
7:6	0x0	CFG2TMC_ATCFG2_DRV_TYPE
3	0x0	CFG2TMC_ATCFG2_SCHMT_EN: ATCFG2 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_ATCFG2_HSM_EN: ATCFG2 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.6 APB_MISC_GP_ATCFG3PADCTRL_0

ATCFG3 Pad Control Register

Offset: 0x878 │ Read/Write: R/W │ Reset: 0x01318000 (0b0000x0010011x0011000xxxx00xx00xx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_ATCFG3_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_ATCFG3_CAL_DRVDN_SLWR
26:20	0x13	CFG2TMC_ATCFG3_CAL_DRVUP
18:12	0x18	CFG2TMC_ATCFG3_CAL_DRVDN
7:6	0x0	CFG2TMC_ATCFG3_DRV_TYPE
3	0x0	CFG2TMC_ATCFG3_SCHMT_EN: ATCFG3 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_ATCFG3_HSM_EN: ATCFG3 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.7 APB_MISC_GP_ATCFG4PADCTRL_0

ATCFG4 Pad Control Register

Offset: 0x87c │ Read/Write: R/W │ Reset: 0x01318000 (0b0000x0010011x0011000xxxx00xx00xx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_ATCFG4_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_ATCFG4_CAL_DRVDN_SLWR
26:20	0x13	CFG2TMC_ATCFG4_CAL_DRVUP
18:12	0x18	CFG2TMC_ATCFG4_CAL_DRVDN
7:6	0x3	CFG2TMC_ATCFG4_DRV_TYPE
3	0x0	CFG2TMC_ATCFG4_SCHMT_EN: ATCFG4 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_ATCFG4_HSM_EN: ATCFG4 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.8 APB_MISC_GP_ATCFG5PADCTRL_0

ATCFG5 Pad Control Register

Offset: 0x880 │ Read/Write: R/W │ Reset: 0xf0b48000 (0b1111xxxx1011010010xxxxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_ATCFG5_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_ATCFG5_CAL_DRVDN_SLWR

Bit	Reset	Description
23:19	0x16	CFG2TMC_ATCFG5_CAL_DRVUP
18:14	0x12	CFG2TMC_ATCFG5_CAL_DRVDN
3	0x0	CFG2TMC_ATCFG5_SCHMT_EN: ATCFG5 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_ATCFG5_HSM_EN: ATCFG5 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.9 APB_MISC_GP_CDEV1CFGPADCTRL_0

CDEV1CFG Pad Control Register

Offset: 0x884 │ Read/Write: R/W │ Reset: 0xf0e09000 (0b1111xxx01110xxx01001xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_CDEV1CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_CDEV1CFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_CDEV1CFG_CAL_DRVUP
16:12	0x9	CFG2TMC_CDEV1CFG_CAL_DRVDN
3	0x0	CFG2TMC_CDEV1CFG_SCHMT_EN: CDEV1CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_CDEV1CFG_HSM_EN: CDEV1CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.10 APB_MISC_GP_CDEV2CFGPADCTRL_0

CDEV2CFG Pad Control Register

Offset: 0x888 │ Read/Write: R/W │ Reset: 0xf0e09000 (0b1111xxx01110xxx01001xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_CDEV2CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_CDEV2CFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_CDEV2CFG_CAL_DRVUP
16:12	0x9	CFG2TMC_CDEV2CFG_CAL_DRVDN
3	0x0	CFG2TMC_CDEV2CFG_SCHMT_EN: CDEV2CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_CDEV2CFG_HSM_EN: CDEV2CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.11 APB_MISC_GP_DAP1CFGPADCTRL_0

DAP1CFG Pad Control Register

Offset: 0x890 | Read/Write: R/W | Reset: 0xf0e09000 (0b1111xxx01110xxx01001xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_DAP1CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DAP1CFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_DAP1CFG_CAL_DRVUP
16:12	0x9	CFG2TMC_DAP1CFG_CAL_DRVDN
3	0x0	CFG2TMC_DAP1CFG_SCHMT_EN: DAP1CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DAP1CFG_HSM_EN: DAP1CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.12 APB_MISC_GP_DAP2CFGPADCTRL_0

DAP2CFG Pad Control Register

Offset: 0x894 | Read/Write: R/W | Reset: 0xf0e09000 (0b1111xxx01110xxx01001xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_DAP2CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DAP2CFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_DAP2CFG_CAL_DRVUP
16:12	0x9	CFG2TMC_DAP2CFG_CAL_DRVDN
3	0x0	CFG2TMC_DAP2CFG_SCHMT_EN: DAP2CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DAP2CFG_HSM_EN: DAP2CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.13 APB_MISC_GP_DAP3CFGPADCTRL_0

DAP3CFG Pad Control Register

Offset: 0x898 | Read/Write: R/W | Reset: 0xf0e09000 (0b1111xxx01110xxx01001xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_DAP3CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DAP3CFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_DAP3CFG_CAL_DRVUP
16:12	0x9	CFG2TMC_DAP3CFG_CAL_DRVDN

Bit	Reset	Description
3	0x0	CFG2TMC_DAP3CFG_SCHMT_EN: DAP3CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DAP3CFG_HSM_EN: DAP3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.14 APB_MISC_GP_DAP4CFGPADCTRL_0

DAP4CFG Pad Control Register

Offset: 0x89c │ Read/Write: R/W │ Reset: 0xf0e09000 (0b1111xxx01110xxx01001xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_DAP4CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DAP4CFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_DAP4CFG_CAL_DRVUP
16:12	0x9	CFG2TMC_DAP4CFG_CAL_DRVDN
3	0x0	CFG2TMC_DAP4CFG_SCHMT_EN: DAP4CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DAP4CFG_HSM_EN: DAP4CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.15 APB_MISC_GP_DBGCFGPADCTRL_0

DBGCFG Pad Control Register

Offset: 0x8a0 │ Read/Write: R/W │ Reset: 0xf0e09000 (0b1111xxx01110xxx01001xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_DBGCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DBGCFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_DBGCFG_CAL_DRVUP
16:12	0x9	CFG2TMC_DBGCFG_CAL_DRVDN
3	0x0	CFG2TMC_DBGCFG_SCHMT_EN: DBGCFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DBGCFG_HSM_EN: DBGCFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.16 APB_MISC_GP_SDIO3CFGPADCTRL_0

SDIO3CFG Pad Control Register

Controls for BDSDMEM pads of SDMMC3

The following calibration codes need to be used for the pad under default condition. In general, the calibration pad will provide the code for the pad. To bypass calibration and provide the default code for the required impedance, these values (in decimal) should be used.

Table 61: DRV Codes

Supply	33 ohms		50 ohms		66 ohms		100 ohms	
	Up	DN	Up	DN	Up	DN	Up	DN
1.8V	70	54	42	32	34	23	19	
3.3V			36	20				

SDIO3CFG Pad Control Register

Offset: 0x8b0 | Read/Write: R/W | Reset: 0x12414000 (0b0001x0100100x0010100xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_SDIO3CFG_CAL_DRVUP_SLWF. Maps to MSB of DRVDN[18:17].
29:28	0x1	CFG2TMC_SDIO3CFG_CAL_DRVDN_SLWR. Maps to MSB of DRVUP[26:25].
26:20	0x24	CFG2TMC_SDIO3CFG_CAL_DRVUP. 3.3V 50 ohm driver for removable SD card.
18:12	0x14	CFG2TMC_SDIO3CFG_CAL_DRVDN. 3.3V 50 ohm driver for removable SD card.
3	0x0	CFG2TMC_SDIO3CFG_SCHMT_EN: SDIO3CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_SDIO3CFG_HSM_EN: SDIO3CFG data pins high speed mode enable. Software should set this if I/O is running at greater than or equal to 100 MHz. 0 = DISABLE 1 = ENABLE

18.1.3.17 APB_MISC_GP_SPICFGPADCTRL_0

SPICFG Pad Control Register

Offset: 0x8b4 | Read/Write: R/W | Reset: 0xf0e09000 (0b1111xxx01110xxx01001xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_SPICFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_SPICFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_SPICFG_CAL_DRVUP
16:12	0x9	CFG2TMC_SPICFG_CAL_DRVDN
3	0x0	CFG2TMC_SPICFG_SCHMT_EN: SPICFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	CFG2TMC_SPICFG_HSM_EN: SPICFG data pins high-speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.18 APB_MISC_GP_UAACFGPADCTRL_0

UAACFG Pad Control Register

Offset: 0x8b8 │ Read/Write: R/W │ Reset: 0xf0e09000 (0b1111xxx01110xxx01001xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_UAACFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_UAACFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_UAACFG_CAL_DRVUP
16:12	0x9	CFG2TMC_UAACFG_CAL_DRVDN
3	0x0	CFG2TMC_UAACFG_SCHMT_EN: UAACFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_UAACFG_HSM_EN: UAACFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.19 APB_MISC_GP_UABCFGPADCTRL_0

UABCFG Pad Control Register

Offset: 0x8bc │ Read/Write: R/W │ Reset: 0xf0e09000 (0b1111xxx01110xxx01001xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_UABCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_UABCFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_UABCFG_CAL_DRVUP
16:12	0x9	CFG2TMC_UABCFG_CAL_DRVDN
3	0x0	CFG2TMC_UABCFG_SCHMT_EN: UABCFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_UABCFG_HSM_EN: UABCFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.20 APB_MISC_GP_UART2CFGPADCTRL_0

UART2CFG Pad Control Register

Offset: 0x8c0 │ Read/Write: R/W │ Reset: 0xf0e09000 (0b1111xxx01110xxx01001xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_UART2CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_UART2CFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_UART2CFG_CAL_DRVUP
16:12	0x9	CFG2TMC_UART2CFG_CAL_DRVDN
3	0x0	CFG2TMC_UART2CFG_SCHMT_EN: UART2CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_UART2CFG_HSM_EN: UART2CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.21 APB_MISC_GP_UART3CFGPADCTRL_0

UART3CFG Pad Control Register

Offset: 0x8c4 │ Read/Write: R/W │ Reset: 0xf0e09000 (0b1111xxx01110xxx01001xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_UART3CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_UART3CFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_UART3CFG_CAL_DRVUP
16:12	0x9	CFG2TMC_UART3CFG_CAL_DRVDN
3	0x0	CFG2TMC_UART3CFG_SCHMT_EN: UART3CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_UART3CFG_HSM_EN: UART3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.22 APB_MISC_GP_SDIO1CFGPADCTRL_0

Controls for BDSDMEM pads of SDMMC1

The following calibration codes need to be used for the pad under default condition. In general, the calibration pad will provide the code for the pad. To bypass calibration and provide the default code for the required impedance, these values (in decimal) should be used.

Table 62: DRV Codes

Supply	33 ohms		50 ohms		66 ohms		100 ohms	
	Up	DN	Up	DN	Up	DN	Up	DN
1.8V	70	54	42	32	34	23	19	
3.3V			36	20				

SDIO1CFG Pad Control Register

Offset: 0x8ec │ Read/Write: R/W │ Reset: 0x52a20000 (0b0101x0101010x0100000xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x1	CFG2TMC_SDIO1CFG_CAL_DRVUP_SLWF. Maps to MSB of DRVDN[18:17]
29:28	0x1	CFG2TMC_SDIO1CFG_CAL_DRVDN_SLWR. Maps to MSB of DRVUP[26:25]
26:20	0x2a	CFG2TMC_SDIO1CFG_CAL_DRVUP. 1.8V 50 ohm driver for wifi SDIO card.
18:12	0x20	CFG2TMC_SDIO1CFG_CAL_DRVDN. 1.8V 50 ohm driver for wifi SDIO card
3	0x0	CFG2TMC_SDIO1CFG_SCHMT_EN: SDIO1CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_SDIO1CFG_HSM_EN: SDIO1CFG data pins high speed mode enable. Software should set this if I/O is running at greater than or equal to 100 MHz. 0 = DISABLE 1 = ENABLE

18.1.3.23 APB_MISC_GP_DDCCFGPADCTRL_0

CRTCFG and DDCCFG Pad Control Registers

Offset: 0x8fc │ Read/Write: R/W │ Reset: 0xc160000c (0b1100xxx10110xxx00000xxxxxxxx11xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_DDCCFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_DDCCFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_DDCCFG_CAL_DRVUP
16:12	0x0	CFG2TMC_DDCCFG_CAL_DRVDN
3	0x1	CFG2TMC_DDCCFG_SCHMT_EN: Data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x1	CFG2TMC_DDCCFG_HSM_EN: Data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.24 APB_MISC_GP_GMACFGPADCTRL_0

Controls for BDSDMEMLV pads of SDMMC4

The following calibration codes need to be used for the pad under default condition. In general, the calibration pad will provide the code for the pad. To bypass calibration and provide the default code for the required impedance, these values (in hexadecimal) should be used.

Table 63: DRV Codes (5-bit Code)

Supply	33 ohms		50 ohms	
	Up	DN	Up	DN
1.8V	0x7	0x7	0x2	0x1

Supply	33 ohms		50 ohms	
1.2V	0xE	0xE	0x5	0x5

GMACFG Pad Control Register

Offset: 0x900 │ Read/Write: R/W │ Reset: 0x00204040 (0b0000xxx00010x00001xxxxxx01xx00x0)

Bit	Reset	Description
31:30	0x0	CFG2TMC_GMACFG_CAL_DRVUP_SLWF. Maps to MSB of DRVDN[18:17].
29:28	0x0	CFG2TMC_GMACFG_CAL_DRVDN_SLWR. Maps to MSB of DRVUP[24:23].
24:20	0x2	CFG2TMC_GMACFG_CAL_DRVUP. 1.8V 50 ohm driver for eMMC.
18:14	0x1	CFG2TMC_GMACFG_CAL_DRVDN. 1.8V 50 ohm driver for eMMC.
7:6	0x1	CFG2TMC_GMACFG_DRV_TYPE. 0x1: 33-50 ohm driver 0x0: 66-100 ohm driver
3	0x0	CFG2TMC_GMACFG_SCHMT_EN: Data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GMACFG_HSM_EN: Data pins high speed mode enable. Software should set this when I/O is running at greater than or equal to 100 MHz. 0 = DISABLE 1 = ENABLE
0	0x0	CFG2TMC_GMACFG_E_PREEMP:pre-emphasis enable 0 = DISABLE 1 = ENABLE

18.1.3.25 APB_MISC_GP_GMECFGPADCTRL_0

GMECFG Pad Control Register

Offset: 0x910 │ Read/Write: R/W │ Reset: 0xf0724000 (0b1111xxxx0111001001xxxxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_GMECFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_GMECFG_CAL_DRVDN_SLWR
23:19	0xe	CFG2TMC_GMECFG_CAL_DRVUP
18:14	0x9	CFG2TMC_GMECFG_CAL_DRVDN
3	0x0	CFG2TMC_GMECFG_SCHMT_EN: Data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GMECFG_HSM_EN: Data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.26 APB_MISC_GP_GMFCFGPADCTRL_0

GMFCFG Pad Control Register

Offset: 0x914 │ Read/Write: R/W │ Reset: 0xf0724000 (0b1111xxxx0111001001xxxxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_GMFCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_GMFCFG_CAL_DRVDN_SLWR
23:19	0xe	CFG2TMC_GMFCFG_CAL_DRVUP
18:14	0x9	CFG2TMC_GMFCFG_CAL_DRVDN
3	0x0	CFG2TMC_GMFCFG_SCHMT_EN: Data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GMFCFG_HSM_EN: Data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.27 APB_MISC_GP_GMGCFGPADCTRL_0

GMGCFG Pad Control Register

Offset: 0x918 │ Read/Write: R/W │ Reset: 0xf0724000 (0b1111xxxx0111001001xxxxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_GMGCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_GMGCFG_CAL_DRVDN_SLWR
23:19	0xe	CFG2TMC_GMGCFG_CAL_DRVUP
18:14	0x9	CFG2TMC_GMGCFG_CAL_DRVDN
3	0x0	CFG2TMC_GMGCFG_SCHMT_EN: Data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GMGCFG_HSM_EN: Data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.28 APB_MISC_GP_GMHCFGPADCTRL_0

GMHCFG Pad Control Register

Offset: 0x91c │ Read/Write: R/W │ Reset: 0xf0724000 (0b1111xxxx0111001001xxxxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_GMHCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_GMHCFG_CAL_DRVDN_SLWR
23:19	0xe	CFG2TMC_GMHCFG_CAL_DRVUP
18:14	0x9	CFG2TMC_GMHCFG_CAL_DRVDN
3	0x0	CFG2TMC_GMHCFG_SCHMT_EN: Data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GMHCFG_HSM_EN: Data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.29 APB_MISC_GP_OWRCFGPADCTRL_0

OWRCFG Pad Control Register

Offset: 0x920 | Read/Write: R/W | Reset: 0xc160000c (0b1100xxx10110xxx00000xxxxxxxx11xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_OWRCFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_OWRCFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_OWRCFG_CAL_DRVUP
16:12	0x0	CFG2TMC_OWRCFG_CAL_DRVDN
3	0x1	CFG2TMC_OWRCFG_SCHMT_EN: Data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x1	CFG2TMC_OWRCFG_HSM_EN: Data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.30 APB_MISC_GP_UADCFGPADCTRL_0

UDACFG Pad Control Register

Offset: 0x924 | Read/Write: R/W | Reset: 0xf0e09000 (0b1111xxx01110xxx01001xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_UDACFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_UDACFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_UDACFG_CAL_DRVUP
16:12	0x9	CFG2TMC_UDACFG_CAL_DRVDN
3	0x0	CFG2TMC_UDACFG_SCHMT_EN: Data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_UDACFG_HSM_EN: Data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.31 APB_MISC_GP_GPVCFGPADCTRL_0

GPV Pad Control Register

Offset: 0x928 | Read/Write: R/W | Reset: 0xf0e09000 (0b1111xxx01110xxx01001xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_GPVCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_GPVCFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_GPVCFG_CAL_DRVUP
16:12	0x9	CFG2TMC_GPVCFG_CAL_DRVDN

Bit	Reset	Description
3	0x0	CFG2TMC_GPVCFG_SCHMT_EN: 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GPVCFG_HSM_EN: 0 = DISABLE 1 = ENABLE

18.1.3.32 APB_MISC_GP_DEV3CFGPADCTRL_0

DEV3CFG Pad Control Register

Offset: 0x92c │ Read/Write: R/W │ Reset: 0xf0e09000 (0b1111xxx01110xxx01001xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_DEV3CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DEV3CFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_DEV3CFG_CAL_DRVUP
16:12	0x9	CFG2TMC_DEV3CFG_CAL_DRVDN
3	0x0	CFG2TMC_DEV3CFG_SCHMT_EN: DEV3CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DEV3CFG_HSM_EN: DEV3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.33 APB_MISC_GP_CECCFGPADCTRL_0

CECCFG Pad Control Register

Offset: 0x938 │ Read/Write: R/W │ Reset: 0xf1612000 (0b1111xxx10110xxx10010xxxxxxxx00xx)

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
3	0x0	CFG2TMC_CECCFG_SCHMT_EN: Data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_CECCFG_HSM_EN: Data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.34 APB_MISC_GP_ATCFG6PADCTRL_0

ATCFG6 Pad Control Register

Offset: 0x994 │ Read/Write: R/W │ Reset: 0x01318000 (0b0000x0010011x0011000xxxx00xx00xx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_ATCFG6_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_ATCFG6_CAL_DRVDN_SLWR
26:20	0x13	CFG2TMC_ATCFG6_CAL_DRVUP
18:12	0x18	CFG2TMC_ATCFG6_CAL_DRVDN
7:6	0x0	CFG2TMC_ATCFG6_DRV_TYPE
3	0x0	CFG2TMC_ATCFG6_SCHMT_EN: ATCFG6 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_ATCFG6_HSM_EN: ATCFG6 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.35 APB_MISC_GP_DAP5CFGPADCTRL_0

DAP5CFG Pad Control Register

Offset: 0x998 │ Read/Write: R/W │ Reset: 0xf0503000 (0b1111xxx00101xxx00011xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_DAP5CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DAP5CFG_CAL_DRVDN_SLWR
24:20	0x5	CFG2TMC_DAP5CFG_CAL_DRVUP
16:12	0x3	CFG2TMC_DAP5CFG_CAL_DRVDN
3	0x0	CFG2TMC_DAP5CFG_SCHMT_EN: DAP5CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DAP5CFG_HSM_EN: DAP5CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.36 APB_MISC_GP_USB_VBUS_EN_CFGPADCTRL_0

USB_VBUS_EN_CFG Pad Control Register

Offset: 0x99c │ Read/Write: R/W │ Reset: 0xf1612000 (0b1111xxx10110xxx10010xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_USB_VBUS_EN_CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_USB_VBUS_EN_CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_USB_VBUS_EN_CFG_CAL_DRVUP
16:12	0x12	CFG2TMC_USB_VBUS_EN_CFG_CAL_DRVDN
3	0x0	CFG2TMC_USB_VBUS_EN_CFG_SCHMT_EN: data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_USB_VBUS_EN_CFG_HSM_EN: Data pins high speed mode enable

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE

18.1.3.37 APB_MISC_GP_AOCFG3PADCTRL_0

AOCFG3 Pad Control Register

Offset: 0x9a8 | Read/Write: R/W | Reset: 0x0000000c (0bxx00xxxxxxxxxx00000xxxxxxxx11xx)

Bit	Reset	Description
29:28	0x0	CFG2TMC_AOCFG3_CAL_DRVDN_SLWR
16:12	0x0	CFG2TMC_AOCFG3_CAL_DRVDN
3	0x1	CFG2TMC_AOCFG3_SCHMT_EN: AOCFG3 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x1	CFG2TMC_AOCFG3_HSM_EN: AOCFG3 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.38 APB_MISC_GP_AOCFG0PADCTRL_0

AOCFG0 Pad Control Register

Offset: 0x9b0 | Read/Write: R/W | Reset: 0xf0e09000 (0b1111xxx01110xxx01001xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_AOCFG0_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_AOCFG0_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_AOCFG0_CAL_DRVUP
16:12	0x9	CFG2TMC_AOCFG0_CAL_DRVDN
3	0x0	CFG2TMC_AOCFG0_SCHMT_EN: AOCFG0 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_AOCFG0_HSM_EN: AOCFG0 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.39 APB_MISC_GP_HVCFG0PADCTRL_0

HVCFG0 Pad Control Register

Offset: 0x9b4 | Read/Write: R/W | Reset: 0x0000000c (0bxx00xxxxxxxxxx00000xxxxxxxx11xx)

Bit	Reset	Description
29:28	0x0	CFG2TMC_HVCFG0_CAL_DRVDN_SLWR
16:12	0x0	CFG2TMC_HVCFG0_CAL_DRVDN
3	0x1	CFG2TMC_HVCFG0_SCHMT_EN: HVCFG0 data pins Schmitt enable 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
2	0x1	CFG2TMC_HVCFG0_HSM_EN: HVCFG0 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.40 APB_MISC_GP_SDIO4CFGPADCTRL_0

SDIO4CFG Pad Control Register

Offset: 0x9c4 | Read/Write: R/W | Reset: 0xc0e0000c (0b1100xxx01110xxx00000xxxxxxxx11xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_SDIO4CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_SDIO4CFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_SDIO4CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_SDIO4CFG_CAL_DRVDN
3	0x1	CFG2TMC_SDIO4CFG_SCHMT_EN: SDIO4CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x1	CFG2TMC_SDIO4CFG_HSM_EN: SDIO4CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.3.41 APB_MISC_GP_AOCFG4PADCTRL_0

Offset: 0x9c8 | Read/Write: R/W | Reset: 0xf0e09000 (0b1111x0001110x0001001xxxx00xx00xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_AOCFG4_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_AOCFG4_CAL_DRVDN_SLWR
26:20	0xe	CFG2TMC_AOCFG4_CAL_DRVUP
18:12	0x9	CFG2TMC_AOCFG4_CAL_DRVDN
7:6	0x0	CFG2TMC_AOCFG4_DRV_TYPE
3	0x0	CFG2TMC_AOCFG4_SCHMT_EN: AOCFG4 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_AOCFG4_HSM_EN: AOCFG4 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

18.1.4 DAC/DAP Registers

18.1.4.1 APB_MISC_DAS_DAP_CTRL_SEL_0

DAP Control Register

This is an array of 5 identical register entries; the register fields below apply to each entry.

Offset: 0xc00..0xc13 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
31	0x0	DAP_MS_SEL: This bit is programmed to put a particular DAP in either master or slave mode when two or more DAPs are in bypass mode. 0 = SLAVE 1 = MASTER
30	0x0	DAP_SDATA1_TX_RX: Programs sdata1 in either tx or rx mode when two or more DAPs are in bypass mode. 0 = TX 1 = RX
29	0x0	DAP_SDATA2_RX_TX: Programs sdata2 in either tx or rx mode when two or more DAPs are in bypass mode. 0 = RX 1 = TX
4:0	0x0	DAP_CTRL_SEL: DAP selection bits to select one of the three DACs or one of the five DAPs. 0 = DAC1 1 = DAC2 2 = DAC3 16 = DAP1 17 = DAP2 18 = DAP3 19 = DAP4 20 = DAP5

18.1.4.2 APB_MISC_DAS_DAC_INPUT_DATA_CLK_SEL_0

DAC Input Data Selections

This is an array of 3 identical register entries; the register fields below apply to each entry.

Offset: 0xc40..0xc4b | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
31:28	0x0	DAC_SDATA2_SEL: These bits control the sdata2 input selection for DACs. 0 = DAP1 1 = DAP2 2 = DAP3 3 = DAP4 4 = DAP5
27:24	0x0	DAC_SDATA1_SEL: These bits control the sdata1 input selection for DACs. 0 = DAP1 1 = DAP2 2 = DAP3 3 = DAP4 4 = DAP5
3:0	0x0	DAC_CLK_SEL: These bits control the bit clock and fsync selection for DACs. 0 = DAP1 1 = DAP2 2 = DAP3 3 = DAP4 4 = DAP5

18.1.5 AP Control Registers

Secure Registers

18.1.5.1 APB_MISC_SECURE_REGS_APB_SLAVE_SECURITY_ENABLE_REG0_0

APB Slave Security Enable Register 0

These registers are used to enable and disable secure access of corresponding APB slaves. If set, the corresponding APB slave is accessed only via secure transactions.

Offset: 0xc00 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000x00000000000x000000x)

Bit	Reset	Description
24	0x0	CEC_SECURITY_EN: CEC
23	0x0	ATOMICS_SECURITY_EN: Atomics
22	0x0	LA_SECURITY_EN: LA
21	0x0	HDA_SECURITY_EN: HDA
20	0x0	SATA_SECURITY_EN: SATA
18	0x0	OWR_SECURITY_EN: OWR
17	0x0	SNOR_SECURITY_EN: SNOR
16	0x0	KFUSE_SECURITY_EN: kfuse
15	0x0	FUSE_SECURITY_EN: Fuse
14	0x0	SE_SECURITY_EN: Security Engine
13	0x0	PMC_SECURITY_EN: PMC
12	0x0	KBC_SECURITY_EN: Reserved
11	0x0	RTC_SECURITY_EN: RTC
10	0x0	CSITE_SECURITY_EN: Core Site
9	0x0	MHS_SECURITY_EN: MIPI HSI
8	0x0	PWM_SECURITY_EN: PWFM
6	0x0	DTV_SECURITY_EN: DTV
5	0x0	VFIR_SECURITY_EN: VFIR
4	0x0	AUDIO_SECURITY_EN: Audio Cluster registers
3	0x0	PINMUX_AUX_SECURITY_EN: Pinmux aux registers
2	0x0	SATA_AUX_SECURITY_EN: SATA aux registers
1	0x0	MISC_REGS_SECURITY_EN: PP, SC1x pads and GP registers

18.1.5.2 APB_MISC_SECURE_REGS_APB_SLAVE_SECURITY_ENABLE_REG1_0

APB Slave Security Enable Register 1

Offset: 0xc04 | Read/Write: R/W | Reset: 0x00000000 (0b000000000000xxxx0000xxxx00xxxxxx)

Bit	Reset	Description
31	0x0	I2C6_SECURITY_EN: I2C6
30	0x0	DVC_SECURITY_EN: DVC - I2C5
29	0x0	I2C4_SECURITY_EN: I2C4
28	0x0	I2C3_SECURITY_EN: I2C3
27	0x0	I2C2_SECURITY_EN: I2C2
26	0x0	I2C1_SECURITY_EN: I2C1
25	0x0	SPI6_SECURITY_EN: SPI6
24	0x0	SPI5_SECURITY_EN: SPI5
23	0x0	SPI4_SECURITY_EN: SPI4
22	0x0	SPI3_SECURITY_EN: SPI3
21	0x0	SPI2_SECURITY_EN: SPI2
20	0x0	SPI1_SECURITY_EN: SPI1
15	0x0	UART_D_SECURITY_EN: UARTD
14	0x0	UART_C_SECURITY_EN: UARTC
13	0x0	UART_B_SECURITY_EN: UARTB
12	0x0	UART_A_SECURITY_EN: UARTA
7	0x0	EMC_SECURITY_EN: EMC
6	0x0	MC_SECURITY_EN: MC

18.1.5.3 APB_MISC_SECURE_REGS_APB_SLAVE_SECURITY_ENABLE_REG2_0

APB Slave Security Enable Register 2

Offset: 0xc08 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx000000000xxxx0000)

Bit	Reset	Description
16	0x0	DVFS_SECURITY_EN: DVFS
15	0x0	MIPI_CAL_SECURITY_EN: MIPI cal
14	0x0	XUSB_PADCTL_SECURITY_EN: XUSB_PADCTL
13	0x0	XUSB_DEV_SECURITY_EN: XUSB_dev
12	0x0	XUSB_HOST_SECURITY_EN: XUSB_host
11	0x0	APB2JTAG_SECURITY_EN: APB2JTAG
10	0x0	SOC_THERM_SECURITY_EN: SOC_THERM
9	0x0	DP2_SECURITY_EN: DP2
8	0x0	DDS_SECURITY_EN: DDS
3	0x0	SDMMC4_SECURITY_EN: SDMMC4

Bit	Reset	Description
2	0x0	SDMMC3_SECURITY_EN: SDMMC3
1	0x0	SDMMC2_SECURITY_EN: SDMMC2
0	0x0	SDMMC1_SECURITY_EN: SDMMC1

18.1.6 Pin Mux Selects

Refer to the Multi-Purpose I/O Pins and Pin Multiplexing (Pinmuxing) section in this document for details on the Pinmux_aux registers.

18.2 APB DMA Controller

The APB DMA Controller is placed between the AHB Bus and the APB Bus and is a master on both buses.

The APB DMA Controller is used for block data transfers from a source location to the destination location. The source may be DRAM or IRAM, and the destination location could be devices placed on APB Bus; or vice versa. DMA transfers are done without any processor intervention other than register writes needed to program the parameters for a particular transfer, and accesses needed to handle any interrupts.

The APB DMA Controller has 32 fully programmable channels, which can all transfer data concurrently.

18.2.1 Features

- DMA master for transfers between external/internal memory and peripheral devices on the APB Bus
- Two modes of operation: single transfer (once) or continuous
- Programmable burst sizes of 1, 4, or 8 words
- Maximum transfer size is 1 GB per channel, with the minimum size being one word
- Programmable APB bus widths of 8, 16, and 32 bits
- Separate AHB and APB start addresses
- Per channel trigger and flow control mechanism support
- Channel to channel trigger support, i.e., ability to link up channels to start at the end of another channel's transfer, allowing scattering/gathering of physical memory.
- Interrupt generation at the completion of channel transfer
- Interrupts per channel can be routed to the 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
- Weighted round robin arbitration among channels at burst granularity. The weights for each channel can be set.
- Runs on APB clock. The APB clock generally runs at 1:2:2 or 1:2:3 clock ratios (sclk : hclk : pclk).
- APB DMA secure access feature allows configuration of secure and non-secure areas
- Address wraparound
- Global or individual channel pausing

18.2.2 Functionality

There are 32 channels in APB DMA. An APB DMA channel can transfer specified portions of data from an AHB address space (can be iRAMs or DRAMs on the MC) to an APB address space. APB DMA follows a simple round robin arbitration scheme, starting with channel 0.

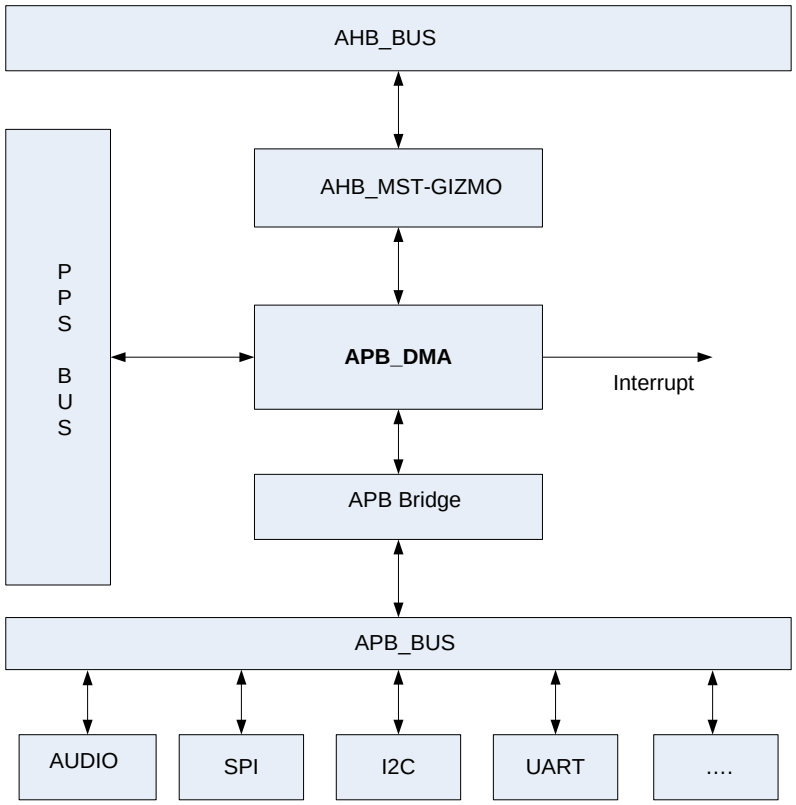
Each channel can have independent burst transfer sizes programmed to one word, four words, or eight words.

Each DMA channel supports:

- Enable bit: one for each channel and one global enable bit.
- Interrupt Enable at the end of transfer with ability to mask or route to desired processor.
- Ability to hold off a processor. The Processor that writes into this bit will be held until transfer is completed.
- Direction bit to determine the direction of transfer--AHB to APB or APB to AHB.

- Trigger and Flow selects. These are addition controls apart from channel enable, on which the transfer depends. Trigger is used to start a channel on some event to start the transfer and flow is used to proceed with every new burst transfer. These events are under either Software or Hardware control.
- Wrap feature wraps the LS nibble of the address back to 'b0000 instead of incrementing to the next address if the required number of words has been transmitted.
- APB bus width is configurable whereas the AHB bus width is fixed to 32 bits. The APB bus can be 8, 16, or 32 bits wide. For an AHB burst of 8 (burst is always with regard to words) and an APB bus size of 16, there are: 8 words transferred to the APB side, or 16 halfwords transferred.
- Double Buffering Mode makes the APB DMA Burst Address reset to AHB Base Address after every even (2nd, 4th, 6th, etc.) APB DMA Transfer. This mode is used only along with continuous mode (once bit 0).
- Separate AHB start address and APB start address.
- Ability to delay the burst rate with the help of a global counter which can be programmed to desired value, which equals number of clocks delay required between each burst.

Figure 36: APB-DMA Block Diagram



18.2.3 APB DMA Secure Access

APB DMA secure access is determined by the channel security enable bits in the APB DMA Security register. A particular channel is secured by enabling the respective bit field . Whether through the CPU or APB DMA, accesses are blocked if the SECURITY_EN bit of that region is enabled and the incoming access is non-secured.

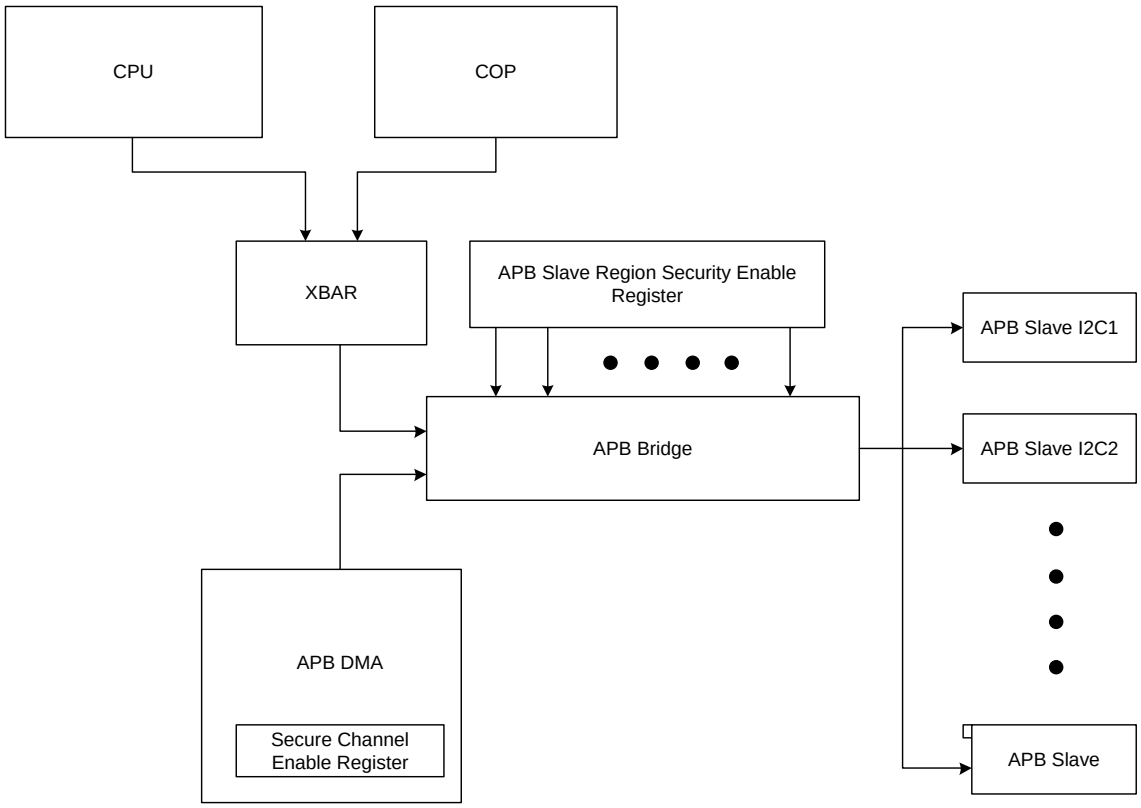
The APBDMA_SECURITY_REG register contains the fields CH_<0-31>_SECURITY_EN, which indicate which channel is secure. Write and read transactions to this register must always be secure.

If a channel is designated as secure, its registers must be written and read out in a secure manner. A non-secure write attempt will not affect any of the secure channel's registers and a non-secure read attempt will not generate any read data.

If a channel is non-secure, its registers can be written or read either in a secure or a non-secure manner.

The following figure shows the security flow.

Figure 37: Security Flow Diagram



18.2.4 Programming Guidelines

Follow these guidelines when programming the APB DMA Controller:

The DMA transfer size should be in multiples of 4 bytes only.

The APB burst size and the FIFO trigger levels (in the modules) need to be programmed such that they do not lead to an overflow/underflow of the FIFO in the APB client.

All registers of a channel should be programmed before the Channel Enable bit in the channel's control register is set as follows:

1. Program the AHB Starting Address and APB Starting address in the *AHB_PTR and *APB_PTR registers.
2. Program the required AHB BURST size, WRAP word window size, and AHB_DATA_SWAP (byte swapping) option in the *AHB_SWQ register. The AHB BUS WIDTH is fixed to a 32-bit bus.
3. Program the required APB_BUS_WIDTH (as the peripheral), APB_DATA_SWAP (byte swapping) option, and WRAP word window size in the *APB_SEQ register.
4. Program the number of words to be transferred in the *WCOUNT register.
5. Program the trigger in the CHANNEL_*_CSRE register.
6. Program the Interrupt option, Hold Processor option, DMA transfer direction, Transfer Mode, and Flow Enable in the CHANNEL_*_CSR register.

7. Program the Global Enable (GEN) bit in the APB_DMA Command Register. The GEN bit can also be programmed before programming the channel registers.
8. Whenever the channel's ENB bit is enabled, the DMA starts the data transfer.
9. Each channel's status is observed by polling the APB_DMA Status Register. The number of words remaining to be transferred will be in the WORD_TRANSFER register.
10. The Tx/Rx Flow/Trigger requestors are programmed in the APB-DMA Requestor Assignments Registers.
11. The busy (BSY) bit is set as soon as a DMA channel is enabled and is cleared after the transfer is completed.

Pausing and Ending of Data Transfers

- Clearing the Global Enable (GEN) bit causes the transfers that are in progress to be paused. Setting the GEN bit resumes the transfers. A channel should never be disabled when the DMA is in global pause.
- Setting the CHANNEL_PAUSE bit in the CSRE register pauses data transfers on that channel. If the CHANNEL_PAUSE bit of a particular channel is enabled during a data transfer, the data transfer is paused only on this channel after the completion of the current burst. It can be resumed by setting this bit to 0.
- If a channel's ENB is disabled independently while a transfer is in progress, the transfer ends after completing any burst sequence that is in progress.

Interrupts

- Interrupts are write-1-to-clear, i.e., an interrupt bit is cleared when the value of write data corresponding to the bit position of the interrupt bit is 1.
- An interrupt is generated when the last data is accepted by the AHB slave while writing on the AHB bus and once the last data is placed on APB bus while reading from AHB bus. Note that completion of an AHB write does not guarantee the data is written to DRAM. Refer to the AHB section of this document for how data is flushed from AHB to the MC for this master.

Wraparound

Wraparound does not occur in between bursts; it is only after completing a burst that the address wraps around. Wraparound is possible for both AHB and APB addresses.

The programming of the AHB wraparound needs to account for the address that is programmed in the AHB start address and the burst size.

Usually, APB clients have a buffer of a fixed size. If the APB addresses are incremented after each word transfer, it is likely that the address would cross the address limit of that client and go into the address range of another client. To avoid this, the wrap feature is used. The default wraparound on the APB side is wrapping on 1 word. It prevents unnecessary address switching on the APB.

With the address wrap feature enabled, the LS nibble of the address wraps back to 0 after completion of the programmed number of words instead of incrementing the address.

Example: AHB burst size = 4 words, AHB start address = 0x4000_0000, AHB wrap on 32 words

In this case, the AHB addresses would be:

0x4000_0000, 0x4000_0004, 0x4000_0008, 0x4000_000C

…….

0x4000_0070, 0x4000_0074, 0x4000_0078, 0x4000_007C

When the address wrap feature is disabled, the addresses are incremented after each word transfer. This helps transfer data from/to contiguous locations.

Flow Control

When flow control is enabled, the number of words to be transferred must always be a multiple of the burst size/APB trigger level. Without flow control the following are possible:

- If a burst of 8 is requested with an address that is not aligned to an 8-word boundary, the DMA will do a single burst until it aligns itself to the 8-word boundary and then starts issuing burst requests.
- 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.
- If a burst of 4 is requested with an address that is not aligned to a 4-word boundary, the DMA transfers a single burst until it aligns itself to the 4-word boundary.
- If a burst of 4 is requested and at any point of time the transfer size goes below 4, the DMA completes the remaining transfers with a burst of 1.

Continuous Mode

In continuous mode, when a transfer is in progress, the APB and AHB addresses and the WCOUNT fields can be reprogrammed in the middle of a data transfer. When the current transfer completes and a new transfer starts, it picks up the newly programmed values. All new programming must complete before the channel completes the data transfer. To be sure about this, CHANNEL_PAUSE can be used.

2x Mode

In 2x mode, the reprogramming can be done only on the second half of the data transfer.

18.2.5 APB DMA Registers

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

18.2.5.1 APBDMA_COMMAND_0

APB-DMA Command Register

The Global Enable (GEN) bit in the command register enables the APB DMA. Clearing this bit causes active DMA transfers to be paused. Pending bus transactions (on going burst) will be completed, and no new transactions will be initiated. Setting this bit again resumes the transfers. Disabling this bit will disable the channel register access.

Note that the power on reset value for the APB DMA Global Enable is 0. This bit must be written to 1 before any APB DMA transactions can begin.

Offset: 0x0 │Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	GEN: Enables Global APB-DMA 0 = DISABLE 1 = ENABLE

18.2.5.2 APBDMA_STATUS_0

APB-DMA Status Register

The Busy bits in the Status Register indicate which (if any) of the APB DMA channels have active pending APB DMA transfers. Note that Busy bits remain active in APB DMA transfers that are started in continuous (repetitive) mode until the enable bit for the APB DMA channel is cleared to 0 (by the software). Read-only flags are set/cleared by hardware.

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

Bit	Reset	Description
31	X	BSY_31: DMA channel 31 status 0 = NOT_BUSY 1 = BUSY
30	X	BSY_30: DMA channel 30 status 0 = NOT_BUSY 1 = BUSY
29	X	BSY_29: DMA channel 29 status 0 = NOT_BUSY 1 = BUSY
28	X	BSY_28: DMA channel 28 status 0 = NOT_BUSY 1 = BUSY
27	X	BSY_27: DMA channel 27 status 0 = NOT_BUSY 1 = BUSY
26	X	BSY_26: DMA channel 26 status 0 = NOT_BUSY 1 = BUSY
25	X	BSY_25: DMA channel 25 status 0 = NOT_BUSY 1 = BUSY
24	X	BSY_24: DMA channel 24 status 0 = NOT_BUSY 1 = BUSY
23	X	BSY_23: DMA channel 23 status 0 = NOT_BUSY 1 = BUSY
22	X	BSY_22: DMA channel 22 status 0 = NOT_BUSY 1 = BUSY
21	X	BSY_21: DMA channel 21 status 0 = NOT_BUSY 1 = BUSY
20	X	BSY_20: DMA channel 20 status 0 = NOT_BUSY 1 = BUSY
19	X	BSY_19: DMA channel 19 status 0 = NOT_BUSY 1 = BUSY
18	X	BSY_18: DMA channel 18 status 0 = NOT_BUSY 1 = BUSY
17	X	BSY_17: DMA channel 17 status 0 = NOT_BUSY 1 = BUSY
16	X	BSY_16: DMA channel 16 status 0 = NOT_BUSY 1 = BUSY
15	X	BSY_15: DMA channel 15 status 0 = NOT_BUSY 1 = BUSY

Bit	Reset	Description
14	X	BSY_14: DMA channel 14 status 0 = NOT_BUSY 1 = BUSY
13	X	BSY_13: DMA channel 13 status 0 = NOT_BUSY 1 = BUSY
12	X	BSY_12: DMA channel 12 status 0 = NOT_BUSY 1 = BUSY
11	X	BSY_11: DMA channel 11 status 0 = NOT_BUSY 1 = BUSY
10	X	BSY_10: DMA channel 10 status 0 = NOT_BUSY 1 = BUSY
9	X	BSY_9: DMA channel 9 status 0 = NOT_BUSY 1 = BUSY
8	X	BSY_8: DMA channel 8 status 0 = NOT_BUSY 1 = BUSY
7	X	BSY_7: DMA channel 7 status 0 = NOT_BUSY 1 = BUSY
6	X	BSY_6: DMA channel 6 status 0 = NOT_BUSY 1 = BUSY
5	X	BSY_5: DMA channel 5 status 0 = NOT_BUSY 1 = BUSY
4	X	BSY_4: DMA channel 4 status 0 = NOT_BUSY 1 = BUSY
3	X	BSY_3: DMA channel 3 status 0 = NOT_BUSY 1 = BUSY
2	X	BSY_2: DMA channel 2 status 0 = NOT_BUSY 1 = BUSY
1	X	BSY_1: DMA channel 1 status 0 = NOT_BUSY 1 = BUSY
0	X	BSY_0: DMA channel 0 status 0 = NOT_BUSY 1 = BUSY

18.2.5.3 APBDMA_CNTRL_REG_0

APB-DMA Counter Register

The APB DMA Counter is used to slow down the request rates on some APB DMA channels. The APB DMA Channel Counter Enable register stores bits that configure which (if any) channel(s) should be throttled (bits [31:0] of channel counter enable register correspond to the 32 APB DMA channels).

The APB DMA Counter initial/reload count value is programmed in the APB DMA Counter Register (bits[15:0]).The APB DMA Counter is loaded with this initial/reload count value whenever the APB DMA Counter Register is written, and is re-loaded to this saved initial count value on a burst complete (programmable). The APB DMA Counter value will decrement whenever an APB DMA burst complete, and the current count value is non-zero, and any of the bits [31:16] are set.

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	COUNT_VALUE: DMA COUNT Value.

18.2.5.4 APBDMA_IRQ_STA_CPU_0

APB-DMA CPU IRQ STATUS Register

Gathers all the after-masking CPU directed IRQ status bits

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

Bit	Reset	Description
31	X	CH31: Gathers all the after-masking CPU directed IRQ status bits from channel 31 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
30	X	CH30: Gathers all the after-masking CPU directed IRQ status bits from channel 30 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
29	X	CH29: Gathers all the after-masking CPU directed IRQ status bits from channel 29 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
28	X	CH28: Gathers all the after-masking CPU directed IRQ status bits from channel 28 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
27	X	CH27: Gathers all the after-masking CPU directed IRQ status bits from channel 27 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
26	X	CH26: Gathers all the after-masking CPU directed IRQ status bits from channel 26 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
25	X	CH25: Gathers all the after-masking CPU directed IRQ status bits from channel 25 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
24	X	CH24: Gathers all the after-masking CPU directed IRQ status bits from channel 24 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
23	X	CH23: Gathers all the after-masking CPU directed IRQ status bits from channel 23 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
22	X	CH22: Gathers all the after-masking CPU directed IRQ status bits from channel 22 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
21	X	CH21: Gathers all the after-masking CPU directed IRQ status bits from channel 21 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
20	X	CH20: Gathers all the after-masking CPU directed IRQ status bits from channel 20 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
19	X	CH19: Gathers all the after-masking CPU directed IRQ status bits from channel 19 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
18	X	CH18: Gathers all the after-masking CPU directed IRQ status bits from channel 18 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
17	X	CH17: Gathers all the after-masking CPU directed IRQ status bits from channel 17 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
16	X	CH16: Gathers all the after-masking CPU directed IRQ status bits from channel 16 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
15	X	CH15: Gathers all the after-masking CPU directed IRQ status bits from channel 15 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
14	X	CH14: Gathers all the after-masking CPU directed IRQ status bits from channel 14 0 = No IRQ pending

Bit	Reset	Description
		1 = IRQ pending 0 = DISABLE 1 = ENABLE
13	X	CH13: Gathers all the after-masking CPU directed IRQ status bits from channel 13 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
12	X	CH12: Gathers all the after-masking CPU directed IRQ status bits from channel 12 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
11	X	CH11: Gathers all the after-masking CPU directed IRQ status bits from channel 11 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
10	X	CH10: Gathers all the after-masking CPU directed IRQ status bits from channel 10 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
9	X	CH9: Gathers all the after-masking CPU directed IRQ status bits from channel 9 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
8	X	CH8: Gathers all the after-masking CPU directed IRQ status bits from channel 8 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
7	X	CH7: Gathers all the after-masking CPU directed IRQ status bits from channel 7 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
6	X	CH6: Gathers all the after-masking CPU directed IRQ status bits from channel 6 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
5	X	CH5: Gathers all the after-masking CPU directed IRQ status bits from channel 5 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
4	X	CH4: Gathers all the after-masking CPU directed IRQ status bits from channel 4 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
3	X	CH3: Gathers all the after-masking CPU directed IRQ status bits from channel 3 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
2	X	CH2: Gathers all the after-masking CPU directed IRQ status bits from channel 2 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
1	X	CH1: Gathers all the after-masking CPU directed IRQ status bits from channel 1 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
0	X	CH0: Gathers all the after-masking CPU directed IRQ status bits from channel 0 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE

18.2.5.5 APBDMA_IRQ_STA_COP_0

APB-DMA COP IRQ STATUS Register

Gathers all the after-masking COP directed IRQ status bits

Offset: 0x18 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CH31: Gathers all the after-masking COP directed IRQ status bits from channel 31 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
30	X	CH30: Gathers all the after-masking COP directed IRQ status bits from channel 30 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
29	X	CH29: Gathers all the after-masking COP directed IRQ status bits from channel 29 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
28	X	CH28: Gathers all the after-masking COP directed IRQ status bits from channel 28 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
27	X	CH27: Gathers all the after-masking COP directed IRQ status bits from channel 27 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
26	X	CH26: Gathers all the after-masking COP directed IRQ status bits from channel 26 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
25	X	CH25: Gathers all the after-masking COP directed IRQ status bits from channel 25 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
24	X	CH24: Gathers all the after-masking COP directed IRQ status bits from channel 24 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
23	X	CH23: Gathers all the after-masking COP directed IRQ status bits from channel 23 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
22	X	CH22: Gathers all the after-masking COP directed IRQ status bits from channel 22 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
21	X	CH21: Gathers all the after-masking COP directed IRQ status bits from channel 21 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
20	X	CH20: Gathers all the after-masking COP directed IRQ status bits from channel 20 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
19	X	CH19: Gathers all the after-masking COP directed IRQ status bits from channel 19 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
18	X	CH18: Gathers all the after-masking COP directed IRQ status bits from channel 18 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
17	X	CH17: Gathers all the after-masking COP directed IRQ status bits from channel 17 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
16	X	CH16: Gathers all the after-masking COP directed IRQ status bits from channel 16 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
15	X	CH15: Gathers all the after-masking COP directed IRQ status bits from channel 15 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
14	X	CH14: Gathers all the after-masking COP directed IRQ status bits from channel 14 0 = No IRQ pending

Bit	Reset	Description
		1 = IRQ pending 0 = DISABLE 1 = ENABLE
13	X	CH13: Gathers all the after-masking COP directed IRQ status bits from channel 13 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
12	X	CH12: Gathers all the after-masking COP directed IRQ status bits from channel 12 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
11	X	CH11: Gathers all the after-masking COP directed IRQ status bits from channel 11 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
10	X	CH10: Gathers all the after-masking COP directed IRQ status bits from channel 10 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
9	X	CH9: Gathers all the after-masking COP directed IRQ status bits from channel 9 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
8	X	CH8: Gathers all the after-masking COP directed IRQ status bits from channel 8 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
7	X	CH7: Gathers all the after-masking COP directed IRQ status bits from channel 7 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
6	X	CH6: Gathers all the after-masking COP directed IRQ status bits from channel 6 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
5	X	CH5: Gathers all the after-masking COP directed IRQ status bits from channel 5 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
4	X	CH4: Gathers all the after-masking COP directed IRQ status bits from channel 4 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
3	X	CH3: Gathers all the after-masking COP directed IRQ status bits from channel 3 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
2	X	CH2: Gathers all the after-masking COP directed IRQ status bits from channel 2 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
1	X	CH1: Gathers all the after-masking COP directed IRQ status bits from channel 1 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
0	X	CH0: Gathers all the after-masking COP directed IRQ status bits from channel 0 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE

18.2.5.6 APBDMA_IRQ_MASK_0

APB-DMA IRQ MASK Register

Allows the IRQ to propagate when enabled, this is the ANDed result of set and clear.

Offset: 0x1c │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CH31: Each bit allows the associated channel31 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
30	X	CH30: Each bit allows the associated channel30 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
29	X	CH29: Each bit allows the associated channel29 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
28	X	CH28: Each bit allows the associated channel28 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
27	X	CH27: Each bit allows the associated channel27 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
26	X	CH26: Each bit allows the associated channel26 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
25	X	CH25: Each bit allows the associated channel25 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
24	X	CH24: Each bit allows the associated channel24 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
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

Bit	Reset	Description
		1 = ENABLE
21	X	CH21: Each bit allows the associated channel21 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
20	X	CH20: Each bit allows the associated channel20 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
19	X	CH19: Each bit allows the associated channel19 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
18	X	CH18: Each bit allows the associated channel18 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
17	X	CH17: Each bit allows the associated channel17 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
16	X	CH16: Each bit allows the associated channel16 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
15	X	CH15: Each bit allows the associated channel15 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
14	X	CH14: Each bit allows the associated channel14 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
13	X	CH13: Each bit allows the associated channel13 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
12	X	CH12: Each bit allows the associated channel12 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
11	X	CH11: Each bit allows the associated channel11 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
10	X	CH10: Each bit allows the associated channel10 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
9	X	CH9: Each bit allows the associated channel9 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
8	X	CH8: Each bit allows the associated channel8 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
7	X	CH7: Each bit allows the associated channel7 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
6	X	CH6: Each bit allows the associated channel6 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
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

Bit	Reset	Description
		1 = ENABLE
3	X	CH3: Each bit allows the associated channel3 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
2	X	CH2: Each bit allows the associated channel2 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
1	X	CH1: Each bit allows the associated channel1 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
0	X	CH0: Each bit allows the associated channel0 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE

18.2.5.7 APBDMA_IRQ_MASK_SET_0

APB-DMA IRQ MASK SET Register

Offset: 0x20 │ Read/Write: WO │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	CH31: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
30	0x0	CH30: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
29	0x0	CH29: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
28	0x0	CH28: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
27	0x0	CH27: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
26	0x0	CH26: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
25	0x0	CH25: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
24	0x0	CH24: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
23	0x0	CH23: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
22	0x0	CH22: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
21	0x0	CH21: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
20	0x0	CH20: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
19	0x0	CH19: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
18	0x0	CH18: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
17	0x0	CH17: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
16	0x0	CH16: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
15	0x0	CH15: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
14	0x0	CH14: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
13	0x0	CH13: Sets the Mask Register 0 = Disable the IRQ

Bit	Reset	Description
		1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
12	0x0	CH12: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
11	0x0	CH11: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
10	0x0	CH10: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
9	0x0	CH9: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
8	0x0	CH8: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
7	0x0	CH7: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
6	0x0	CH6: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
5	0x0	CH5: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
4	0x0	CH4: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
3	0x0	CH3: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
2	0x0	CH2: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
1	0x0	CH1: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
0	0x0	CH0: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE

18.2.5.8 APBDMA_IRQ_MASK_CLR_0

APB-DMA IRQ MASK CLEAR Register

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

Bit	Reset	Description
31	0x0	CH31: Clears the Mask Register 0 = DISABLE 1 = ENABLE
30	0x0	CH30: Clears the Mask Register 0 = DISABLE 1 = ENABLE
29	0x0	CH29: Clears the Mask Register 0 = DISABLE 1 = ENABLE
28	0x0	CH28: Clears the Mask Register 0 = DISABLE 1 = ENABLE
27	0x0	CH27: Clears the Mask Register 0 = DISABLE 1 = ENABLE
26	0x0	CH26: Clears the Mask Register 0 = DISABLE 1 = ENABLE
25	0x0	CH25: Clears the Mask Register 0 = DISABLE 1 = ENABLE
24	0x0	CH24: Clears the Mask Register 0 = DISABLE 1 = ENABLE
23	0x0	CH23: Clears the Mask Register 0 = DISABLE 1 = ENABLE
22	0x0	CH22: Clears the Mask Register 0 = DISABLE 1 = ENABLE
21	0x0	CH21: Clears the Mask Register 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	0x0	CH20: Clears the Mask Register 0 = DISABLE 1 = ENABLE
19	0x0	CH19: Clears the Mask Register 0 = DISABLE 1 = ENABLE
18	0x0	CH18: Clears the Mask Register 0 = DISABLE 1 = ENABLE
17	0x0	CH17: Clears the Mask Register 0 = DISABLE 1 = ENABLE
16	0x0	CH16: Clears the Mask Register 0 = DISABLE 1 = ENABLE
15	0x0	CH15: Clears the Mask Register 0 = DISABLE 1 = ENABLE
14	0x0	CH14: Clears the Mask Register 0 = DISABLE 1 = ENABLE
13	0x0	CH13: Clears the Mask Register 0 = DISABLE 1 = ENABLE
12	0x0	CH12: Clears the Mask Register 0 = DISABLE 1 = ENABLE
11	0x0	CH11: Clears the Mask Register 0 = DISABLE 1 = ENABLE
10	0x0	CH10: Clears the Mask Register 0 = DISABLE 1 = ENABLE
9	0x0	CH9: Clears the Mask Register 0 = DISABLE 1 = ENABLE
8	0x0	CH8: Clears the Mask Register 0 = DISABLE 1 = ENABLE
7	0x0	CH7: Clears the Mask Register 0 = DISABLE 1 = ENABLE
6	0x0	CH6: Clears the Mask Register 0 = DISABLE 1 = ENABLE
5	0x0	CH5: Clears the Mask Register 0 = DISABLE 1 = ENABLE
4	0x0	CH4: Clears the Mask Register 0 = DISABLE 1 = ENABLE
3	0x0	CH3: Clears the Mask Register 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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

18.2.5.9 APBDMA_TRIG_REG_0

APB-DMA Trigger Register

Offset: 0x28 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8	X	TMR2: Trigger select from Timer (Hardware initiated DMA request) 0 = NOT_ACTIVE 1 = ACTIVE
7	X	TMR1: Trigger select from Timer (Hardware initiated DMA request) 0 = NOT_ACTIVE 1 = ACTIVE
6	X	XRQ_B: XRQ.B (GPIOB) (Hardware initiated DMA request) 0 = NOT_ACTIVE 1 = ACTIVE
5	X	XRQ_A: XRQ.A (GPIOA) (Hardware initiated DMA request) 0 = NOT_ACTIVE 1 = ACTIVE
4	X	SMP_27: Semaphore requests software-initiated DMA request 0 = NOT_ACTIVE 1 = ACTIVE
3	X	SMP_26: Semaphore requests software-initiated DMA request 0 = NOT_ACTIVE 1 = ACTIVE
2	X	SMP_25: Semaphore requests software-initiated DMA request 0 = NOT_ACTIVE 1 = ACTIVE
1	X	SMP_24: Semaphore requests software-initiated DMA request 0 = NOT_ACTIVE 1 = ACTIVE

18.2.5.10 APBDMA_CHANNEL_TRIG_REG_0

APB-DMA Channel Trigger Registers

Offset: 0x2c │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	APB_31: EOC-31 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE

Bit	Reset	Description
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 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

Bit	Reset	Description
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 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

18.2.5.11 APBDMA_DMA_STATUS_0

APB-DMA Interrupt Status

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

Bit	Reset	Description
31	X	ISE_EOC_31: DMA Channel31 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
30	X	ISE_EOC_30: DMA Channel30 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE

Bit	Reset	Description
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
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

Bit	Reset	Description
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
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

18.2.5.12 APBDMA_CHANNEL_EN_REG_0

APB-DMA Channel Counter Enable Registers

Offset: 0x34 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	CH31_CNT_EN: Enable the Channel31 count 0 = DISABLE 1 = ENABLE
30	0x0	CH30_CNT_EN: Enable the Channel30 count 0 = DISABLE 1 = ENABLE
29	0x0	CH29_CNT_EN: Enable the Channel29 count 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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 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

Bit	Reset	Description
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
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

18.2.5.13 APBDMA_SECURITY_REG_0

Security enables for each channel.

Secure: TrustZone Protected

Offset: 0x38 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	CH_31_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
30	0x0	CH_30_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
29	0x0	CH_29_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
28	0x0	CH_28_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
27	0x0	CH_27_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
26	0x0	CH_26_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
25	0x0	CH_25_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
24	0x0	CH_24_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
23	0x0	CH_23_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
22	0x0	CH_22_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
21	0x0	CH_21_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
20	0x0	CH_20_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
19	0x0	CH_19_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
18	0x0	CH_18_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
17	0x0	CH_17_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
16	0x0	CH_16_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
15	0x0	CH_15_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
14	0x0	CH_14_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
13	0x0	CH_13_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
12	0x0	CH_12_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
11	0x0	CH_11_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
10	0x0	CH_10_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
9	0x0	CH_9_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
8	0x0	CH_8_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
7	0x0	CH_7_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
6	0x0	CH_6_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
5	0x0	CH_5_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
4	0x0	CH_4_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
3	0x0	CH_3_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
2	0x0	CH_2_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
1	0x0	CH_1_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
0	0x0	CH_0_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE

18.2.5.14 APBDMA_CHANNEL_SWID_0

SWID[0] for each APBDMA channel. Supports secure writes.

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

Bit	Reset	Description
31	0x0	CH_31_SWID:SWID for Channel 31
30	0x0	CH_30_SWID:SWID for Channel 30
29	0x0	CH_29_SWID:SWID for Channel 29
28	0x0	CH_28_SWID:SWID for Channel 28
27	0x0	CH_27_SWID:SWID for Channel 27
26	0x0	CH_26_SWID:SWID for Channel 26
25	0x0	CH_25_SWID:SWID for Channel 25
24	0x0	CH_24_SWID:SWID for Channel 24
23	0x0	CH_23_SWID:SWID for Channel 23
22	0x0	CH_22_SWID:SWID for Channel 22
21	0x0	CH_21_SWID:SWID for Channel 21
20	0x0	CH_20_SWID:SWID for Channel 20

Bit	Reset	Description
19	0x0	CH_19_SWID:SWID for Channel 19
18	0x0	CH_18_SWID:SWID for Channel 18
17	0x0	CH_17_SWID:SWID for Channel 17
16	0x0	CH_16_SWID:SWID for Channel 16
15	0x0	CH_15_SWID:SWID for Channel 15
14	0x0	CH_14_SWID:SWID for Channel 14
13	0x0	CH_13_SWID:SWID for Channel 13
12	0x0	CH_12_SWID:SWID for Channel 12
11	0x0	CH_11_SWID:SWID for Channel 11
10	0x0	CH_10_SWID:SWID for Channel 10
9	0x0	CH_9_SWID:SWID for Channel 9
8	0x0	CH_8_SWID:SWID for Channel 8
7	0x0	CH_7_SWID:SWID for Channel 7
6	0x0	CH_6_SWID:SWID for Channel 6
5	0x0	CH_5_SWID:SWID for Channel 5
4	0x0	CH_4_SWID:SWID for Channel 4
3	0x0	CH_3_SWID:SWID for Channel 3
2	0x0	CH_2_SWID:SWID for Channel 2
1	0x0	CH_1_SWID:SWID for Channel 1
0	0x0	CH_0_SWID:SWID for Channel 0

18.2.5.15 APBDMA_CHAN_WT_REG0_0

Channel weights for weighted-round-robin arbitration

Offset: 0x44 | Read/Write: R/W | Reset: 0x11111111 (0b00010001000100010001000100010001)

Bit	Reset	Description
31:28	0x1	WT_CH31: Weight of channel
27:24	0x1	WT_CH30: Weight of channel
23:20	0x1	WT_CH29: Weight of channel
19:16	0x1	WT_CH28: Weight of channel
15:12	0x1	WT_CH27: Weight of channel
11:8	0x1	WT_CH26: Weight of channel
7:4	0x1	WT_CH25: Weight of channel
3:0	0x1	WT_CH24: Weight of channel

18.2.5.16 APBDMA_CHAN_WT_REG1_0

Channel weights for weighted-round-robin arbitration

Offset: 0x48 | Read/Write: R/W | Reset: 0x11111111 (0b00010001000100010001000100010001)

Bit	Reset	Description

Bit	Reset	Description
31:28	0x1	WT_CH23: Weight of channel
27:24	0x1	WT_CH22: Weight of channel
23:20	0x1	WT_CH21: Weight of channel
19:16	0x1	WT_CH20: Weight of channel
15:12	0x1	WT_CH19: Weight of channel
11:8	0x1	WT_CH18: Weight of channel
7:4	0x1	WT_CH17: Weight of channel
3:0	0x1	WT_CH16: Weight of channel

18.2.5.17 APBDMA_CHAN_WT_REG2_0

Channel weights for weighted-round-robin arbitration

Offset: 0x4c | Read/Write: R/W | Reset: 0x11111111 (0b00010001000100010001000100010001)

Bit	Reset	Description
31:28	0x1	WT_CH15: Weight of channel
27:24	0x1	WT_CH14: Weight of channel
23:20	0x1	WT_CH13: Weight of channel
19:16	0x1	WT_CH12: Weight of channel
15:12	0x1	WT_CH11: Weight of channel
11:8	0x1	WT_CH10: Weight of channel
7:4	0x1	WT_CH9: Weight of channel
3:0	0x1	WT_CH8: Weight of channel

18.2.5.18 APBDMA_CHAN_WT_REG3_0

Channel weights for weighted-round-robin arbitration

Offset: 0x50 | Read/Write: R/W | Reset: 0x11111111 (0b00010001000100010001000100010001)

Bit	Reset	Description
31:28	0x1	WT_CH7: Weight of channel
27:24	0x1	WT_CH6: Weight of channel
23:20	0x1	WT_CH5: Weight of channel
19:16	0x1	WT_CH4: Weight of channel
15:12	0x1	WT_CH3: Weight of channel
11:8	0x1	WT_CH2: Weight of channel
7:4	0x1	WT_CH1: Weight of channel
3:0	0x1	WT_CH0: Weight of channel

18.2.5.19 APBDMA_CHANNEL_SWID1_0

SWID[1] indicating secure ASID. Supports secure writes.

Offset: 0x54 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	CH_31_SWID_1: SWID for Channel 31
30	0x0	CH_30_SWID_1: SWID for Channel 30
29	0x0	CH_29_SWID_1: SWID for Channel 29
28	0x0	CH_28_SWID_1: SWID for Channel 28
27	0x0	CH_27_SWID_1: SWID for Channel 27
26	0x0	CH_26_SWID_1: SWID for Channel 26
25	0x0	CH_25_SWID_1: SWID for Channel 25
24	0x0	CH_24_SWID_1: SWID for Channel 24
23	0x0	CH_23_SWID_1: SWID for Channel 23
22	0x0	CH_22_SWID_1: SWID for Channel 22
21	0x0	CH_21_SWID_1: SWID for Channel 21
20	0x0	CH_20_SWID_1: SWID for Channel 20
19	0x0	CH_19_SWID_1: SWID for Channel 19
18	0x0	CH_18_SWID_1: SWID for Channel 18
17	0x0	CH_17_SWID_1: SWID for Channel 17
16	0x0	CH_16_SWID_1: SWID for Channel 16
15	0x0	CH_15_SWID_1: SWID for Channel 15
14	0x0	CH_14_SWID_1: SWID for Channel 14
13	0x0	CH_13_SWID_1: SWID for Channel 13
12	0x0	CH_12_SWID_1: SWID for Channel 12
11	0x0	CH_11_SWID_1: SWID for Channel 11
10	0x0	CH_10_SWID_1: SWID for Channel 10
9	0x0	CH_9_SWID_1: SWID for Channel 9
8	0x0	CH_8_SWID_1: SWID for Channel 8
7	0x0	CH_7_SWID_1: SWID for Channel 7
6	0x0	CH_6_SWID_1: SWID for Channel 6
5	0x0	CH_5_SWID_1: SWID for Channel 5
4	0x0	CH_4_SWID_1: SWID for Channel 4
3	0x0	CH_3_SWID_1: SWID for Channel 3
2	0x0	CH_2_SWID_1: SWID for Channel 2
1	0x0	CH_1_SWID_1: SWID for Channel 1
0	0x0	CH_0_SWID_1: SWID for Channel 0

18.2.6 APB DMA Channel Registers

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

Each of the 32 APB DMA channels has its own set of 10 APB DMA registers. Each channel has 64 bytes of address space. This subsection defines one complete set of APB DMA channel registers. The register spaces per APB DMA channel are listed in the table below.

Table 64: APB DMA Channel Register Mapping

Register Space	Channel Registers
0x0 – 0x3f	Channel 0 APB DMA Channel Registers
0x40 – 0x7f	Channel 1 APB DMA Channel Registers
0x80 – 0xbf	Channel 2 APB DMA Channel Registers
0xc0 – 0xff	Channel 3 APB DMA Channel Registers
0x100 – 0x13f	Channel 4 APB DMA Channel Registers
0x140 – 0x17f	Channel 5 APB DMA Channel Registers
0x180 – 0x1bf	Channel 6 APB DMA Channel Registers
0x1c0 – 0x1ff	Channel 7 APB DMA Channel Registers
0x200 – 0x23f	Channel 8 APB DMA Channel Registers
0x240 – 0x27f	Channel 9 APB DMA Channel Registers
0x280 – 0x2bf	Channel 10 APB DMA Channel Registers
0x2c0 – 0x2ff	Channel 11 APB DMA Channel Registers
0x300 – 0x33f	Channel 12 APB DMA Channel Registers
0x340 – 0x37f	Channel 13 APB DMA Channel Registers
0x380 – 0x3bf	Channel 14 APB DMA Channel Registers
0x3c0 – 0x3ff	Channel 15 APB DMA Channel Registers
0x400 – 0x43f	Channel 16 APB DMA Channel Registers
0x440 – 0x47f	Channel 17 APB DMA Channel Registers
0x480 – 0x4bf	Channel 18 APB DMA Channel Registers
0x4c0 – 0x4ff	Channel 19 APB DMA Channel Registers
0x500 – 0x53f	Channel 20 APB DMA Channel Registers
0x540 – 0x57f	Channel 21 APB DMA Channel Registers
0x580 – 0x5bf	Channel 22 APB DMA Channel Registers
0x5c0 – 0x5ff	Channel 23 APB DMA Channel Registers
0x600 – 0x63f	Channel 24 APB DMA Channel Registers
0x640 – 0x67f	Channel 25 APB DMA Channel Registers
0x680 – 0x6bf	Channel 26 APB DMA Channel Registers
0x6c0 – 0x6ff	Channel 27 APB DMA Channel Registers
0x700 – 0x73f	Channel 28 APB DMA Channel Registers
0x740 – 0x77f	Channel 29 APB DMA Channel Registers
0x780 – 0x7bf	Channel 30 APB DMA Channel Registers
0x7c0 – 0x7ff	Channel 31 APB DMA Channel Registers

18.2.6.1 APBDMACHAN_CHANNEL_n_CSR_0

APB-DMA-n Control Register

There are 32 APB DMA Channel Control Registers, one per channel (n = 0 through 31).

Writing a 1 to bit [31] of an APB DMA Channel Control Register will initiate the APB DMA transfer. Because this action will depend on some values programmed in the other registers, it is recommended that the registers in the address space in the required APB DMA channel be programmed before writing into control register.

In "Once" mode, the channel is disabled after the APB DMA transfer has completed. When the channel's transfer completes, an interrupt will be sent if the IE.EOC bit is set. If the transfer requires a trigger or flow, then the corresponding trigger selected by the "TRIG_SEL" or "REQ_SEL" field must become active respectively before the channel can starts its transfer. If both Trigger and Flow bits are set, both conditions must be met before the channel starts each DMA burst.

Offset: 0x0+ (n * 0x40) │ Read/Write: R/W │ Reset: 0x00000000 (0b00000xxxxx000000xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	ENB: Enables DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Generates 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. 0 = APB read to AHB write 1 = AHB read to APB write. 0 = AHB_WRITE 1 = AHB_READ
27	0x0	ONCE: Run Once or Run Multiple Mode (Allow Retriggering of this Channel). 0 = Run for Multiple Block Transfers 1 = Run for One Block Transfer. 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers). 0 = Independent of DRQ request 1 = Link to DRQ source. 0 = DISABLE 1 = ENABLE
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = APBIF_CH0 2 = APBIF_CH1 3 = APBIF_CH2 4 = APBIF_CH3 5 = HSI 6 = APBIF_CH4 7 = APBIF_CH5 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = APBIF_CH6 13 = APBIF_CH7 14 = APBIF_CH8 15 = SL2B1 16 = SL2B2 17 = SL2B3

Bit	Reset	Description
		18 = SL2B4 19 = UART_D 20 = Reserved 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = APBIF_CH9 30 = I2C6 31 = NA31

18.2.6.2 APBDMACHAN_CHANNEL_n_STA_0

APB-DMA-n Status Register

There are 32 APB DMA Status Registers, one per channel (n = 0 through 31).

Offset: 0x4 + (n * 0x40) │ Read/Write: R/W │ Reset: 0xXX000000 (0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RO	X	BSY: Indicates whether DMA Channel Status is active or not 0 = Inactive 1 = 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_STA: • If dir = AHB_WRITE: 0 - Ping buffer transfer completed; 1 - Pong buffer transfer completed. • If dir = AHB_READ: 0 - Pong buffer transfer completed ; 1 - Ping buffer transfer completed ; 0 = PING_INTR_STA 1 = PONG_INTR_STA
27	RO	X	DMA_ACTIVITY: Indicates current DMA channel is transferring data 0 = IDLE 1 = BUSY
26	RO	X	CHANNEL_PAUSE: Indicates the status of channel pause.

18.2.6.3 APBDMACHAN_CHANNEL_n_DMA_BYTE_STA_0

APB-DMA-n DMA Byte Status Register

There are 32 APB DMA Byte Status Registers, one per channel (n = 0 through 31).

Offset: 0x8+ (n * 0x40) │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

18.2.6.4 APBDMACHAN_CHANNEL_n_CSRE_0

APB-DMA-n Control-Extended Register

There are 32 APB DMA Control Extended Registers, one per channel (n = 0 through 31).

Offset: 0xc+ (n * 0x40) │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxxxxxx000000xxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	CHANNEL_PAUSE: When enabled, pauses data transfers on the channel 0 = RESUME 1 = PAUSE
19:14	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 25 = APB_16 26 = APB_17 27 = APB_18 28 = APB_19 29 = APB_20 30 = APB_21 31 = APB_22 32 = APB_23 33 = APB_24 34 = APB_25 35 = APB_26 36 = APB_27 37 = APB_28 38 = APB_29 39 = APB_30 40 = APB_31

18.2.6.5 APBDMACHAN_CHANNEL_n_AHB_PTR_0

APB-DMA-n AHB Starting Address Pointer Register

There are 32 APB Starting Address Pointer Registers, one per channel (n = 0 through 31).

Offset: 0x10 + (n * 0x40) │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000xx)

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: Software writes to modify

18.2.6.6 APBDMACHAN_CHANNEL_n_AHB_SEQ_0

APB-DMA-n AHB Address Sequencer Register

There are 32 APB DMA AHB Address Sequencer Registers, one per channel (n = 0 through 31).

Offset: 0x14 + (x * 0x40) │ Read/Write: R/W │ Reset: 0x26000000 (0b00100110xxxx0000xxxxxxxxxxxxxxxx)

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 (default). 3 = 64 bit-Bus (RSVD). 4 = 128-bit Bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	AHB_DATA_SWAP: When enabled, the data going to AHB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24] }. 0 = DISABLE 1 = ENABLE
26:24	0x6	AHB_BURST: AHB Burst Size DMA Burst Length (encoded). 4 = 1 Word (1x32bits). 5 = 4 Words (4x32 bits). 6 = 8 Words (8x32bits), default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS
19	0x0	DBL_BUF: 2X Double Buffering Mode (for Run-Multiple Mode with No Wrap Operations). 1 = Reload Base Address for 2X blocks (reload every other time). 0 = Reload Base Address for 1X blocks (default) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS
18:16	0x0	WRAP: AHB Address Wrap: AHB Address wrap-around window. 0=No Wrap (default). 5=Wrap on 512 word window. 1=Wrap on 32 word window. 6=Wrap on 1024 word window. 2=Wrap on 64 word window. 7=Wrap on 2048 word window. 3=Wrap on 128 word window. 4=Wrap on 256 word window 0 = NO_WRAP 1 = WRAP_0N_32WORDS 2 = WRAP_ON_64WORDS 3 = WRAP_ON_128WORDS 4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

18.2.6.7 APBDMACHAN_CHANNEL_n_APB_PTR_0

APB-DMA-n APB Starting Address Pointer Register

There are 32 APB DMA APB Starting Address Pointer Registers, one per channel (n = 0 through 31).

Offset: 0x18 + (n * 0x40) │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000xx)

Bit	Reset	Description
31:2	0x0	APB_BASE: APB-DMA Starting address for APB Bus

18.2.6.8 APBDMACHAN_CHANNEL_n_APB_SEQ_0

APB-DMA-n APB Address Sequencer Assignments

There are 32 APB DMA APB Address Sequencer Registers, one per channel (n = 0 through 31).

Offset: 0x1c + (n * 0x40) │ Read/Write: R/W │ Reset: 0x20010000 (0bx0100xxxxxxxx001xxxxxxxxxxxxxxx)

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8-bit bus. 1 = 16-bit bus. 2 = 32-bit bus (default) 3 = 64-bit bus. (RSVD). 4 = 128-bit bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	APB_DATA_SWAP: When enabled, the data going to the APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24] }. 0 = DISABLE 1 = ENABLE
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window. 0 = No Wrap. 1 = Wrap on 1 Word Window (default). 2 = Wrap on 2 Word Window. 3 = Wrap on 4 Word Window. 4 = Wrap on 8 Word Window. 5 = Wrap on 16 Word Window. 6 = Wrap on 32 Word Window. 7 = Wrap on 64 Word Windows (rsvd). 0 = NO_WRAP 1 = WRAP_0N_1WORDS 2 = WRAP_ON_2WORDS 3 = WRAP_ON_4WORDS 4 = WRAP_ON_8WORDS 5 = WRAP_ON_16WORDS 6 = WRAP_ON_32WORDS 7 = WRAP_ON_64WORDS

18.2.6.9 APBDMACHAN_CHANNEL_n_WCOUNT_0

APB-DMA-n Word Count Register

There are 32 APB DMA Word Count Registers, one per channel (n = 0 through 31).

Offset: 0x20 + (n * 0x40) │ Read/Write: R/W │ Reset: 0x00000000 (0bxx00000000000000000000000000xx)

Bit	Reset	Description
29:2	0x0	WCOUNT: Number of 32-bit word cycles.

18.2.6.10 APBDMACHAN_CHANNEL_n_WORD_TRANSFER_0

APB-DMA-n Word Transfer Register

There are 32 APB DMA Word Transfer Registers, one per channel (n = 0 through 31).

Offset: 0x24 + (n * 0x40) │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:2	X	COUNT: APB-Current 32-bit Word Cycles. Flags set/cleared by hardware.

[THIS PAGE INTENTIONALLY LEFT BLANK]

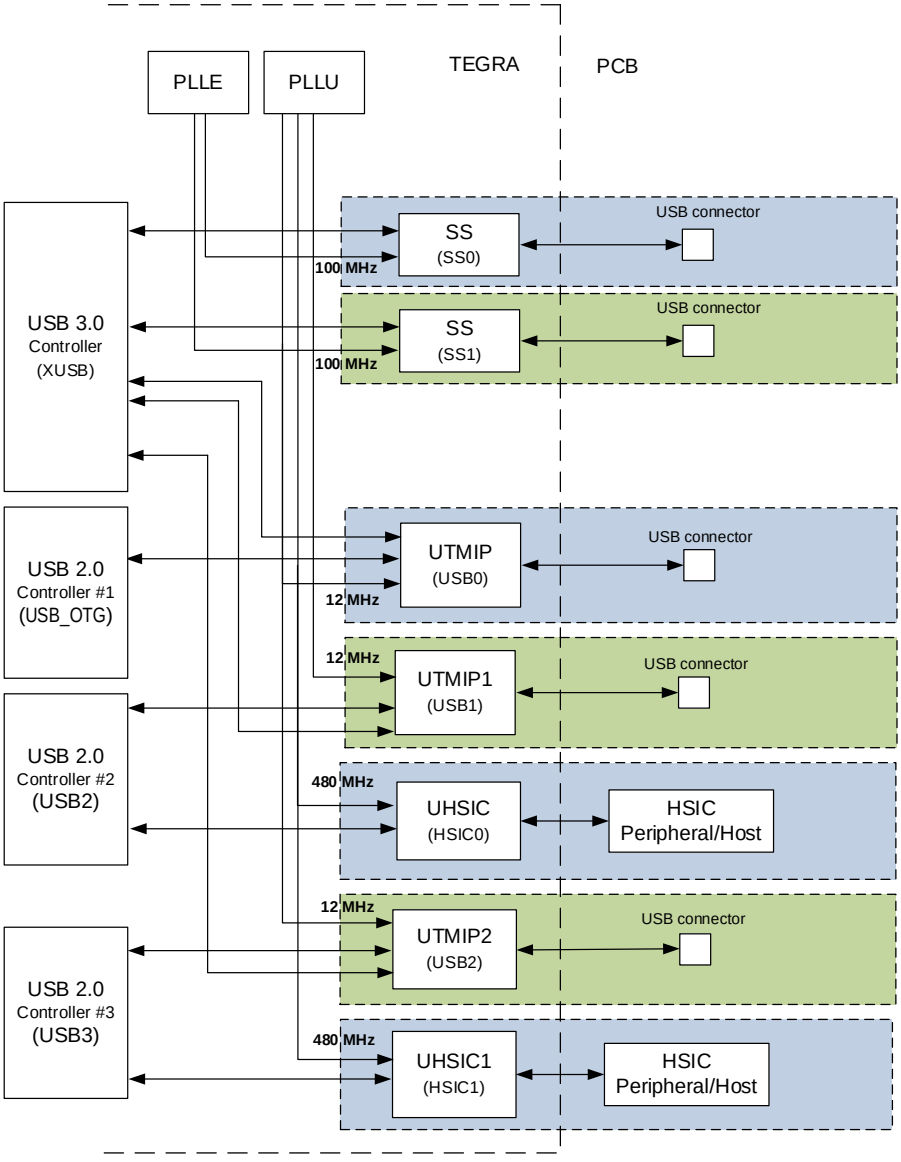
19.0 USB COMPLEX

Note: The ULPI and XUSB device mode functionalities described in this section have been deprecated from Tegra® K1 devices, and are not supported by NVIDIA.

19.1 USB Overview

The Tegra K1 device has three USB 2.0 (EHCI) controllers (named USB_OTG, USB2, and USB3) and one USB 3.0 (xHCI) controller (named XUSB), controlling a total of five exposed ports shared among them (named UTMIP, HSIC0, UTMIP1, HSIC1, and UTMIP2). The following diagram shows the relationship between the different USB controllers and interfaces. Note that some USB ports are shared between the USB 2.0 and USB 3.0 controllers.

Figure 38: Tegra K1 USB Controllers and Interfaces



Below is a summary of the key components in the USB complex:

- USB_OTG and UTMIP: This is the primary USB port that supports OTG. USB recovery is also supported on this interface.
- USB2 and UHSIC: This interface allows connection of an HSIC peripheral/host on the PCB board.
- USB2 and UTMIP1: This interface allows an additional USB port. It supports host mode of operation.
- USB3 and UHSIC1: This interface allows connection of an HSIC peripheral/host on the PCB board.
- USB3 and UTMIP2: This interface allows an additional USB port. It supports host mode of operation.
- XUSB and UTMIP2. XUSB can support multiple interfaces at the same time.
- XUSB and UTMIP1. XUSB can support multiple interfaces at the same time.
- XUSB and UTMIP. XUSB can support multiple interfaces at the same time.
- XUSB and SS. XUSB can support multiple interfaces at the same time.

On power-on-reset, USB2 and USB3 default to UTMI mode.

USB_OTG supports 16 bidirectional endpoints in device mode, where USB2 and USB3 should never be advised or programmed to support device mode. Endpoint 0 of USB_OTG is always a bidirectional control endpoint. All other endpoints can be programmed as one of Bulk, Interrupt, Isochronous, or Control type in either direction. Control endpoints are always bidirectional and hence to program an endpoint as control type, both IN and OUT directions need to be programmed to control type. Conversely, if an endpoint has either IN or OUT direction programmed to be non-control type, the other direction also needs to be programmed to be non-control type. The maximum packet size supported on any endpoint is 1024 bytes in high-speed mode, for both device and host modes.

19.2 USB 2.0 Controllers and Interfaces

The USB 2.0 controllers in the Tegra K1 mobile processors provide mechanisms for the processor to communicate with a PC as a peripheral or to communicate with USB 2.0 peripherals, such as keyboards, mouse devices, cameras, or storage devices, as a host using regular USB 2.0 ports. The USB 2.0 controllers also provide mechanisms for Tegra K1 devices to communicate with an on-board baseband controller with HSIC or regular UTMIP interfaces. Each Tegra K1 device provides three USB 2.0 controllers with five USB interfaces: three regular USB ports and two HSICs (see Figure 38).

The Tegra K1 device supports peripheral functionalities with USB 2.0 controller #1, which implements VBUS and ID sensing capabilities and associated interrupt mechanisms to support for device and host OTG operations with the regular USB 2.0 port. Tegra K1 devices support battery charging charger detection capabilities and associated interrupt mechanisms that are compliant with USB Battery Charging specification version 1.2 with USB 2.0 controller #1. For battery charging, software programming sequences are required to support the operations in response to hardware notifications from USB 2.0 controller #1.

All 3 USB 2.0 controllers support the EHCI programming model for scheduling transactions and interface managements as hosts. Each USB 2.0 controller contains an integrated transaction translator hub to support USB 1.1 transactions when connected to a USB 1.0 peripheral. USB 2.0 controller #1 supports EHCI-like programming models for scheduling responses to host requests.

Tegra K1 USB 2.0 controllers support L1 and L2 (suspend) link power managements. Tegra K1 USB 2.0 controllers support remote wakeup, wake on connect, wake on disconnect, and wake on overcurrent in all Tegra K1 power states, including deep sleep mode.

19.2.1 USB 2.0 Controller #1 (USB_OTG)

USB 2.0 Controller #1 only connects to USB 2.0 Port #1, which is the primary USB 2.0 port on Tegra K1 devices. USB 2.0 Controller #1 shares the same USB 2.0 Port #1 pins with the USB 3.0 xHCI controller. The pinmux must be programmed prior to USB 2.0 Controller #1 using USB 2.0 Port #1.

USB 2.0 Controller #1 supports both USB 2.0 device and USB 2.0 host operations. USB recovery is supported only with USB 2.0 Controller #1 and USB 2.0 Port #1.

Wake-up from deep sleep mode is also supported on VBUS and accessory detect (ACCx_DETECT) pins. Power and Ground pins for USB1 and USB3 are shared.

19.2.2 USB 2.0 Controller #2

USB 2.0 Controller #2 must always be configured to Host mode, and it can be configured to connect to USB2.0 Port #2 or HSIC Port #1.

USB 2.0 Port

USB 2.0 Controller #2 can be configured to use a regular USB 2.0 port. This is the second regular USB 2.0 port on Tegra K1 devices. USB 2.0 Controller #2 shares the same USB 2.0 Port #2 pins with the USB 3.0 controller. The pinmux must be programmed prior to USB 2.0 Controller #2 using USB 2.0 Port #2.

This USB 2.0 port only supports Host mode operations.

HSIC

USB 2.0 Controller #2 can be configured to use HSIC interface #1 that allows connection of an on-board peripheral supporting an HSIC interface to the Tegra K1 device. HSIC can be used to support baseband controllers that support an HSIC interface.

19.2.3 USB 2.0 Controller #3

USB 2.0 Controller #3 must always be configured to Host mode, and it can be configured to connect to USB2.0 Port #2or HSIC Port #2.

USB 2.0 Port

USB 2.0 Controller #3 can be configured to use a regular USB 2.0 port. This is the second regular USB 2.0 port on Tegra K1 devices. USB 2.0 Controller #3 shares the same USB 2.0 Port #3 pins with the USB 3.0 controller. The pinmux must be programmed prior to USB 2.0 Controller #3 using USB 2.0 Port #3.

This USB 2.0 port only supports Host mode operations.

HSIC

USB 2.0 Controller #3 can be configured to use the HSIC #2 interface that allows connection of an on-board peripheral supporting an HSIC interface to the Tegra K1 device. HSIC can be used to support baseband controllers that support an HSIC interface.

19.2.4 Interface Restrictions

Regular 2.0 port and HSIC are mutually exclusive for USB 2.0 controller #2 and #3, where only one of the interfaces can be used with a given board design.

On power-on-reset, the default interface for USB 2.0 controller #2 is the UTMIP interface, and the default interface of USB 2.0 controller #3 is the UTMIP interface.

19.2.5 AHB Interface

All 3 USB 2.0 controllers adapt the AHB interface as masters and slaves for communicating with the other Tegra K1 devices. The following table lists the AHB interface numbers for each controller.

Table 65: AHB Master and Slave Numbers

Controller	AHB Master Number	AHB Slave Number
USB 2.0 Controller #1	6	6
USB 2.0 Controller #2	18	9
USB 2.0 Controller #3	17	11

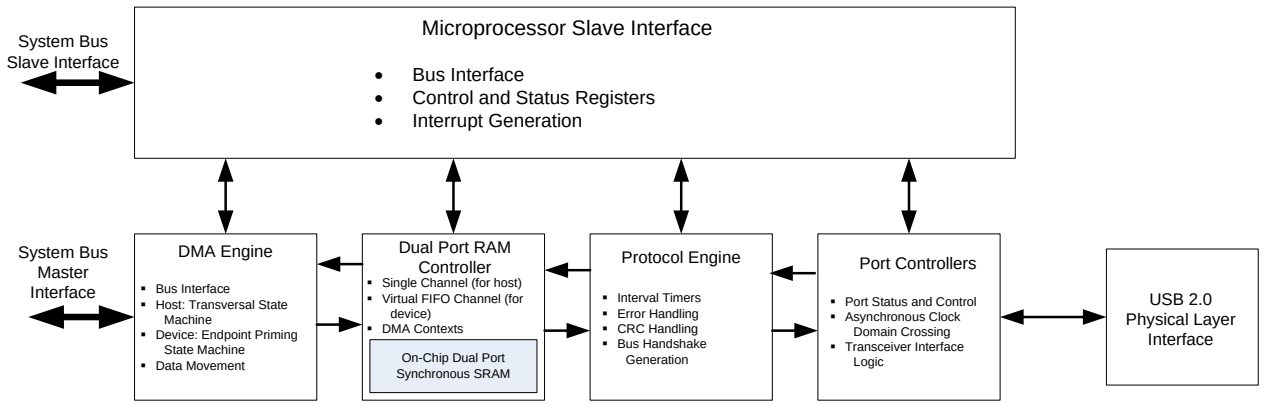
19.3 USB 2.0 Programming Interfaces

All 3 USB 2.0 controllers offer the same functionalities and may be configured to either host or device. However, due to interface restrictions, only USB 2.0 controller #1 should be configured to device mode, while USB 2.0 controller #2 and #3 must always be configured to host mode.

- Host controller registers and data structures are implemented as standard EHCI programming interface.
- Host controller supports USB 1.1 Full and Low speed devices via integrated transaction translator hub through EHCI standard data structures, instead of utilizing companion USB 1.1 host controller.
- Device controller registers and data structures are implemented as extensions to the EHCI programmers interface.

Figure 39 illustrates the control and datapaths block diagram of USB 2.0 controllers.

Figure 39: USB 2.0 Controller Block Diagram



19.3.1 Endpoint Capabilities

USB 2.0 controller #1 supports 16 IN and 16 OUT endpoints in device mode. Endpoint 0 is always a bidirectional control endpoint. All other endpoints can be configured as Bulk, Interrupt, Isochronous or Control endpoints in either direction. Control endpoints are always bidirectional. Hence for a control endpoint, the endpoints in both directions must be configured to be control endpoint. Conversely, if a particular endpoint number has either IN or OUT direction programmed to be a non-control endpoint, the other direction must also be configured as a non-control endpoint.

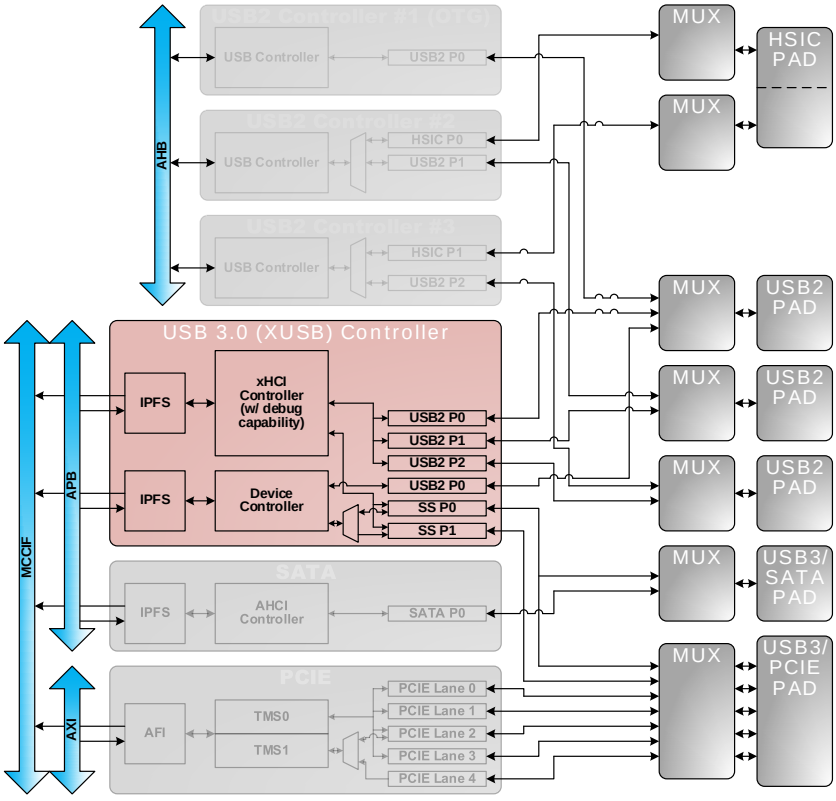
The maximum packet size supported on any endpoint is 1024 bytes in high-speed mode.

19.4 USB 3.0 Controller

The Tegra K1 device provides 1 USB 3.0 controller that can support up to 2 regular USB 3.0 ports and their companion regular USB 2.0 ports, or up to 3 standalone USB 2.0 ports. The USB 3.0 controller in Tegra K1 devices provides mechanisms to communicate with USB 2.0 peripherals, such as keyboards, mouse devices, and card readers. The USB 3.0 controller also provides mechanisms to communicate with USB 3.0 peripherals, such as cameras and storage devices, as a host using regular USB 3.0 ports.

The following figure illustrates the connection between USB 3.0 controllers and USB interfaces in Tegra K1 devices, where each USB 3.0 receptacle must encompass 1 USB 3.0 port and 1 USB 2.0 port from the USB 3.0 controller. USB 3.0 Controller shares the same USB 2.0 Port #1, #2, and #3 pins with USB 2.0 controllers #1, #2, and #3. The pinmux must be programmed prior to USB 3.0 Controller using USB 2.0 Port #1, #2, or #3.

Figure 40: Tegra K1 USB 3.0 Controller and Interfaces



The USB 3.0 controllers support the xHCI programming model for scheduling transactions and interface managements as a host that natively supports USB 3.0, USB 2.0, and USB 1.1 transactions with its USB 3.0 and USB 2.0 interfaces.

The Tegra K1 USB 3.0 controller supports USB 2.0 L1 and L2 (suspend) link power management and USB 3.0 U1, U2, and U3 (suspend) link power managements. The Tegra K1 USB 3.0 controller supports remote wakeup, wake on connect, wake on disconnect, and wake on overcurrent in all Tegra K1 power states, including deep sleep mode.

USB 2.0 Ports

Each USB 2.0 port operates in USB 2.0 High Speed mode when connecting directly to a USB 2.0 peripheral. Each USB 2.0 port operates in USB 1.1 Full and Low Speed modes when connecting directly to a USB 1.1 peripheral.

All USB 2.0 ports operating in High Speed mode share one High Speed Bus Instance, which means 480 Mb/s theoretical bandwidth is distributed across these ports. All USB 2.0 ports operating in Full or Low Speed modes share one Full/Low Speed Bus Instance, which means 12 Mb/s theoretical bandwidth is distributed across these ports.

USB 2.0 ports of the USB 3.0 controller support software initiated L1 and L2 (suspend) link power management. USB 2.0 ports of the USB 3.0 controller do not support hardware initiated L1 link power management.

USB 3.0 Ports

USB 3.0 ports only operate in USB 3.0 Super Speed mode. All USB 3.0 ports share one Super Speed Bus Instance, which means 5 Gb/s theoretical bandwidth is distributed across these ports.

USB 3.0 ports of the USB 3.0 controller support hardware initiated U1 and U2 link power management as well as software initiate U3 (suspend) link power management.

Falcon Microcontroller

The Tegra K1 USB 3.0 controller integrated an NVIDIA Falcon microcontroller to perform the following tasks:

- Command ring processing
- Event ring management
- Endpoint scheduling
- Context save and restore

19.4.1 Interface Restrictions

A USB3.0 connector includes both USB2.0 and USB3.0 interface signals. The USB2.0 interface signals of a USB3.0 connector must be assigned to the USB3.0 controller for xHCI specification compliance.

Similarly, if the USB3.0 interface signals are used to connect to a peripheral on the system board, such as a USB3.0 hub, the USB2.0 signals connecting to that peripheral must be assigned to the USB3.0 controller.

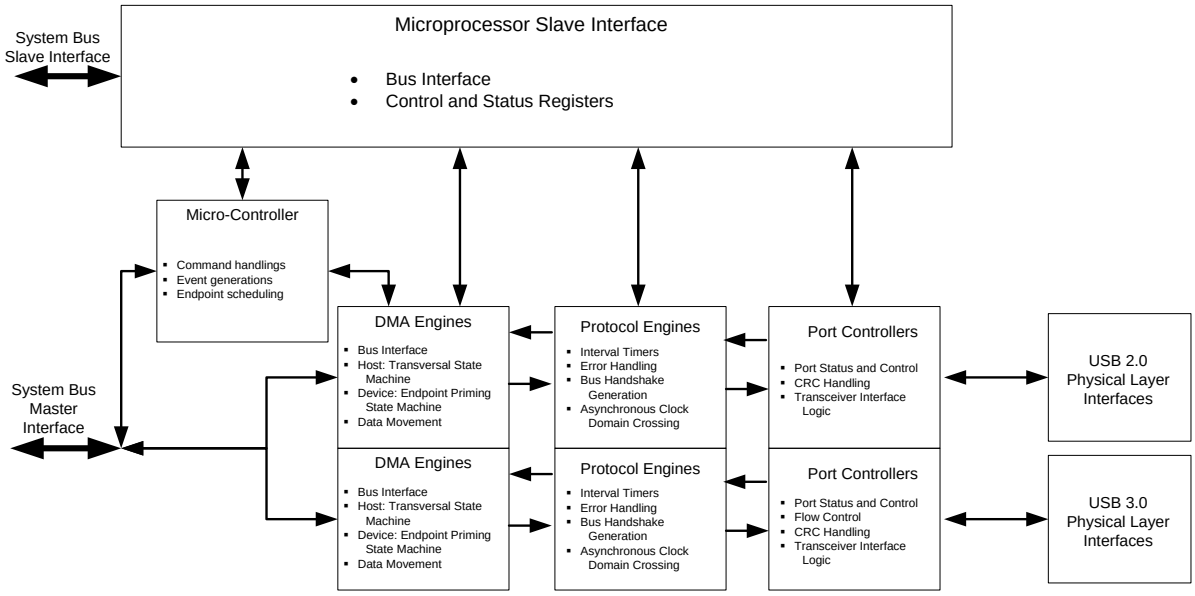
19.4.2 Host and Memory Access Interfaces

USB 3.0 controller adapts the APB interface as its host interface for configuration and memory-mapped I/O register accesses. USB 3.0 controller adapts direct memory as its memory interface for direct memory accesses.

19.5 USB 3.0 Programming Interface

The USB 3.0 Controller supports host functionality with host controller registers and data structures implemented as standard xHCI programming interface. Figure 41 illustrates the control and datapaths block diagram of the USB 3.0 controller.

Figure 41: USB 3.0 Controller Block Diagram



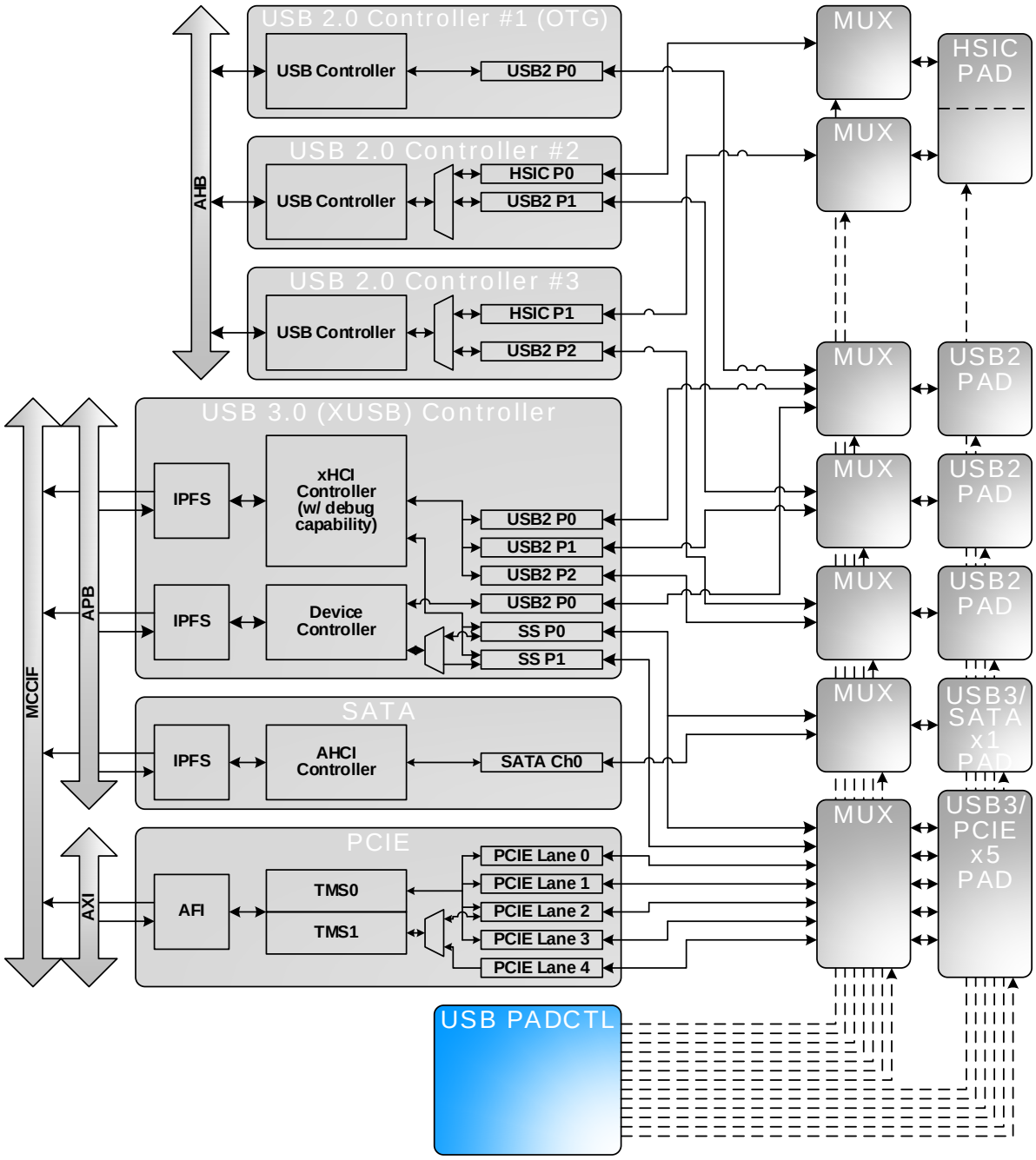
19.6 USB PADCTL

The USB PADCTL in Tegra K1 devices provides the programming interface to assign the USB 2.0 ports to either USB 2.0 controllers or the USB 3.0 controller. The USB PADCTL provides VBUS control and overcurrent mapping for all USB 2.0 controllers and the USB 3.0 controller.

The USB PADCTL also provides the programming interface to configure the pad parameters for the USB 3.0 pad and the USB 2.0 pad for the USB 2.0 port that is assigned to USB 3.0 controller.

The following figure illustrates the connection between USB PADCTL and USB interfaces in a Tegra K1 device.

Figure 42: Tegra K1 USB PADCTL and Interfaces



19.6.1 APB Interface

USB PADCTL adapts the APB interface as its host interface for configuration and memory-mapped I/O register accesses.

19.7 Programming Guidelines

19.7.1 USB_OTG Programming Guidelines

19.7.1.1 USB_OTG Programming

Clock Initialization

After power-on-reset, the USB clocks are disabled. Software can enable the USB clocks by setting CLK_ENB_USBD in the CLK_OUT_ENB_L register to ENABLE. Also, the USB controller stays in reset. Software should bring it out of reset by first setting SWR_USBD_RST in the RST_DEVICES_L register to ENABLE and then setting it to DISABLE.

Some parameters in UTMIP need to be programmed before the UTMIP can be brought out of reset. These parameters need to be programmed every time the USB controller is reset: the SWR_USBD_RST in RST_DEVICES_L register and RST in the USB2D_USBCMD register.

After UTMIP is reset, the PHY clock takes some time to come up, so software must wait until the USB_PHY_CLK_VALID bit in the USB_SUSP_CTRL register becomes valid. This bit, when set to 1, also generates an interrupt if RSM_IE is set in the USB_SUSP_CTRL register.

OTG_ID and VBUS Connection

Tegra K1 devices can use OTG_ID supported by the UTMIP/USB hardware, or a GPIO can be used for OTG_ID functions. If using OTG_ID as a GPIO, the GPIO can be sampled to detect the presence of a micro-A or micro-B plug and the sampled value can be written to the SW_ID bit in the PHY_VBUS_WAKEUP_ID_0 register. It is recommended that board design connects OTG_ID on the Tegra K1 device to the ID pin on the micro-AB connector even if OTG_ID is connected to a GPIO.

Tegra K1 devices have a VBUS pin connected to the 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 directly connect to the VBUS pin. It is required that its voltage is above 3V when VBUS is on.

Both VBUS and ID are available as a wake-up event during DPD/LP0 state. USB bus D+/D- line wake events are also available as a DPD/LP0 state. Refer to the PMC section for more information.

Detection of USB Cable Insertion (General Use Cases)

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 K1 device because of the weak pull-up on this pin. This makes the Tegra K1 start-up as a B-device, even though a cable is not connected (when the ID pin is seen as a 1 by the Tegra K1 device, there can be no-cable or micro-B connected).

When a user attaches the USB cable, one of the following two events occur:

1. The user connects the micro-A end of the cable to the Tegra K1 device. This changes the ID pin to 0. Hence software can detect the ID change and go to A-device mode. Software can then supply power to the USB and place the Tegra K1 device in host mode.
2. The user connects micro-B (or mini-B) end of the cable to the Tegra K1 device. The other end of the cable can be either micro-A or standard-A and is connected to an OTG host or a standard host, respectively. For both cases, the Host starts driving power on VBUS and software can detect the cable insertion event by checking that the voltage on VBUS has gone above the A_Sess_Vld level by reading the A_VBUS_VLD_STS bit in the USB_PHY_VBUS_SENSORS register. Once this is seen, it can place the Tegra K1 device in the device mode.

Notes:

1. ID change can be detected by enabling the GPIO interrupt.

2. VBUS change can be detected by enabling the interrupt A_VBUS_VLD_INT_EN in the USB_PHY_VBUS_SENSORS register.
3. The USB controller and PHY clocks need not be turned on for this detection.
4. To detect these events during DPD/LP0 modes, use VBUS/ID wakeup events in the PMC.

PHY Clock Control

The PHY clock can be turned off using the SUSP_SET bit in the USB_SUSP_CTRL register. This bit must be pulsed by first writing a 1 and then writing a 0 to the bit.

To turn on the PHY clock, software should write to SUSP_CLR in the 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 the USB_PHY_CLK_VALID bit of the USB_SUSP_CTRL register. If this bit is 1, the PHY clock is turned on; otherwise it is off.

The PHY clock can be turned on automatically on the events mentioned below. Set only the required wakeup enable bits based on the state of the USB as mentioned below. Do not set unnecessary wakeup enable bits. When the PHY clock wakes up under any such event, make sure that the wakeup enable bits are turned off as soon as possible.

There is an interrupt generated whenever the PHY clock is turned on which can be checked by checking that the value of USB_PHY_CLK_VALID in the USB_SUSP_CTRL register is 1. The interrupt can be enabled/disabled by setting the RSM_IE bit in USB_SUSP_CTRL to 1/0.

Notes:

1. The PLLU_ENABLE bit in the PLLU_BASE register should be always set to ENABLE. All parameters for PLLU in the PLLU_BASE and PLLU_MISC registers 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.
2. 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 might 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 depending on whether the Tegra K1 processor is in device or host mode.

Device Mode

The PHY clock can be turned off in two cases:

- When the USB cable is not connected, the VBUS signal is 0. In this case, software needs to turn on the PHY clock on cable insertion. Follow the procedure described above in “Detection of USB Cable Insertion”.
- When the USB cable is connected, the PHY clock can only be turned off when the SLI bit in the 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 SLE bit in the USB2D_USBINTR register is 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 the USB_SUSP_CTRL register is set to 1, the PHY clock can be turned on automatically on receiving a USB resume event from the USB Host. If WAKE_ON_DISCON_EN in the USB_SUSP_CTRL register is set to 1, the PHY clock can be turned on automatically on receiving a USB reset event from the 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 the suspend state.

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 resumes 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 in the suspend state by setting the SUSP bit in the USB2D_PORTSC1 register. It needs to wait until this bit is set to 1 by the controller to make sure that the USB system actually went into the suspend state. It can enable the WK_RSM_EN bit in the USB_SUSP_CTRL register and the WK_DS bit in the USB2D_PORTSC1 register. Then it can stop the PHY clock as described above. The PHY clock is turned on automatically if the USB device disconnects or it does a remote wakeup signaling. There is an interrupt associated with this as described above.

If the software wants to wake up the USB system, then it needs to turn on the PHY clock as described above.

Charger Detection 1 (Non-compliant Chargers)

There are four conditions for the different kind of non-compliant chargers: SE1, FS-PU, LS-PU, SE0. Any kind of non-compliant charger can be supported by following the generic procedure below:

1. VBUS detection (by whatever means software does it)
2. Enable USB controller and PHY power and clocks
3. Wait until PHY_CLKVALID = 1.
4. Wait for 10 microseconds more to give time for any filtering to pass through so software does not read the wrong values.
5. Set the DIV_DET_EN bit in the IF_USB_PHY_VBUS_WAKEUP_ID_0 register to 1.
6. Wait for about 10 microseconds to allow the divider detector to settle.
7. Read the {VOP_DIV2P7_DET, VOP_DIV2P0_DET, VON_DIV2P7_DET, VON_DIV2P0_DET} register bits in the IF_USB_PHY_VBUS_WAKEUP_ID_0 register.

   {VOP_DIV2P7_DET, VON_DIV2P7_DET } = 2'b11: Connected to charger_A1.

   {VOP_DIV2P7_DET, VON_DIV2P0_DET } = 2'b11: Connected to charger_A2.

   {VOP_DIV2P0_DET, VON_DIV2P7_DET } = 2'b11: Connected to charger_A3.

   {VOP_DIV2P0_DET, VON_DIV2P0_DET } = 2'b11: Connected to charger_A4.
8. Set the DIV_DET_EN bit in the IF_USB_PHY_VBUS_WAKEUP_ID_0 register to 0.
9. For A1, A2, A3, or A4, enable charging current per the connected charger's requirement (A1, A2, A3, or A4).
10. Read the LS bits [11:10] in the PORTSC register.

    Case 1: LS = 2'b11: Connected to charger_1.

    Case 2: LS = 2'b01: Connected to charger_2.

    Case 3: LS = 2'b10: Connected to charger_3.

    Case 4: LS = 2'b00: Connected to either charger_4, a compliant charger, or a PC host.
11. For case 1, 2, or 3, enable charging current as per that charger's requirement. This can vary for every implementation.

12. For case 4, the connection is to either a PC host or charger_4 (which could be a USB compliant charger as well). Go to step 15.
13. Set the USB circuit to low-power mode now. Make sure the charger circuit does not change state.
14. VBUS disconnect. Return to step 1.
15. Check for connection to a USB compliant charger by doing the charger detection from VBAT register. If yes, go back to step 8. If not, continue to the next step.
16. Set the USB controller to device mode and enable it by setting it to RUN mode. Wait for SOF.
17. If an SOF is not seen after timeout, then the connection is to charger_4. Set the current limit as required for that charger. Go to step 7.

Charger Detection 2 (Compliant Chargers)

This replaces step 10 in the above.

1. Make sure the UTMIP_PD_CHRG bit in the UTMIP_BAT_CHRG_CFG0_0 register is set to 0, so that charger detection circuit is on.
2. Set the UTMIP_OP_I_SRC_EN bit in the UTMIP_BAT_CHRG_CFG0_0 register to 1.
3. Wait for about 10 microseconds to allow the battery charger detector to settle.
4. Now check that the VDCD_DET_CHG_DET bit is 1, and the VDCD_DET_STS bit is 0 in the USB_PHY_VBUS_WAKEUP_ID register. This ensures that the data pins have made contact.
5. Set the UTMIP_OP_I_SRC_EN bit to 0 in the UTMIP_BAT_CHRG_CFG0_0 register.
6. Set the following register bits in the UTMIP_BAT_CHRG_CFG0_0 register to:

   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 that the VDAT_DET_CHG_DET and VDAT_DET_STS bits in the USB_PHY_VBUS_WAKEUP_ID register are set to 1. This indicates connection to a USB compliant dedicated charger that supports up to 1.5A current. If these bits are not set, then the connection is to a PC host, where the downstream port may support charging.
9. Set the following bits in the UTMIP_BAT_CHRG_CFG0_0 register to:

   UTMIP_OP_SRC_EN = 0

   UTMIP_ON_SINK_EN = 0

   UTMIP_OP_SINK_EN = 1

   UTMIP_ON_SRC_EN = 1
10. Wait for about 10 microseconds to allow the battery charger detector to settle.
11. Now check that the VDAT_DET_CHG_DET and VDAT_DET_STS bits in the USB_PHY_VBUS_WAKEUP_ID register are both set to 1. This indicates connection to a USB compliant dedicated charger that supports up to 1.5A current. If these bits are not set, then the connection is to a PC host, where the downstream port supports a standard 500mA current when the high power device is enumerated.

12. Set the following bits in the UTMIP_BAT_CHRG_CFG0_0 register to:

    UTMIP_OP_SRC_EN = 0

    UTMIP_ON_SINK_EN = 0

    UTMIP_OP_SINK_EN = 0

    UTMIP_ON_SRC_EN = 0

13. Now normal USB operation can continue if connected to PC host.

19.7.1.2 LPM Support

Software should program the following register to set the duration that the host keeps generating resume signaling when the host initiated an exit from L1 based on system and device specific requirements:

CONTROLLER_USB2D_USBCMD_0.HIRD

The following software walk-around is required to extend the resume signaling if required.

1. Move PAD out of Suspend

   IF_USB_SUSP_CTRL_0.USB_SUSP_CLR = SET (1)

2. Fake resume signaling

   UTMIP_MISC_CFG0_0.UTMIP_DPDM_OBSERVE = 1

   UTMIP_MISC_CFG0_0.UTMIP_DPDM_OBSERVE_SEL = 0xE

3. Wait the required time following the encoding of the HIRD value

4. Enable hardware to drive the resume signaling

   CONTROLLER_USB2D_PORTSC1_0.FPR = Resume driven on port (1)

5. Disable fake resume signaling as hardware takes over

   UTMIP_MISC_CFG0_0.UTMIP_DPDM_OBSERVE = 0

6. Wait for hardware to finish driving resume signaling by checking the following bit:

   CONTROLLER_USB2D_PORTSC1_0.FPR = No resume driven on port (0)

7. Wait 50 µs

8. Enable hardware to start sending SOF and executing schedule:

   CONTROLLER_USB2D_USBCMD_0.RS = RUN (1)

Software Initiated L1 Support

Software puts the bus into the L1 or Suspend (L2) state by setting the same SUSP bit in PORTSC1 register. The difference is to put the link to L1, software should first program the following registers:

CONTROLLER_USB2D_PORTSC1_0.DA = N, Device Address of the attached device

CONTROLLER_USB2D_HOSTPC1_DEVLC_0. EPLPM = 0

CONTROLLER_USB2D_PORTSC1_0.SLP = Suspend using L1 (1)

Software should then set the following register bit to initiate L1 entry:

CONTROLLER_USB2D_PORTSC1_0.SUSP = 1

Software can check the following registers to ensure the link has entered L1:

CONTROLLER_USB2D_PORTSC1_0.SUSP = Port in suspend state (1)

CONTROLLER_USB2D_PORTSC1_0.SSTS = L1STATE_ENTERED (0)

If L1 cannot be entered successfully, the bus stays in the L0 state. Hardware will generate a port change interrupt when L1 entry fails. So software can check the following registers for the reason of the failed entry. Software can retry the L1 entry later if the reason is NYET_PERIPH, where the peripheral indicates it is not ready to put the bus into L1.

CONTROLLER_USB2D_PORTSC1_0.SUSP = Port not in suspend state (0)

CONTROLLER_USB2D_PORTSC1_0.SSTS = NYET_PERIPH (1), or

= L1STATE_NOT_SUPPORTED (2),

= PERIPH_NORESP_ERR (3),

When the bus is in the L1, software can set the following register at any time to bring the link back to the L0 state.

CONTROLLER_USB2D_PORTSC1_0.FPR = Resume driven on port (1)

Hardware Initiated L1 Support

Software can enable hardware to put the bus to L1 automatically. Software should first program the following registers, where *X* is the system specific idle threshold measured in number of SOFs with no activities in between:

CONTROLLER_USB2D_PORTSC1_0.DA = N, device address of the attached device

CONTROLLER_USB2D_HOSTPC1_DEVLC_0. EPLPM = 0

CONTROLLER_USB2D_HOSTPC1_DEVLC_0. LPMFRM = X

Software should then set the following register bit to enabled hardware initiated L1 entry:

CONTROLLER_USB2D_HOSTPC1_DEVLC_0. LPMX = 1

19.7.1.3 Streaming Mode

The USB controller supports streaming, where it starts sending DATA packets for OUT transactions when the controller has not finished fetching the entire data packet from system memory to its TX buffer. For IN transactions, the streaming mode allows data to be written to system memory before the entire data packet is received in its RX buffer. Streaming mode can be disabled with the following bit:

CONTROLLER_USB2D_USBMODE_0.SDIS

Enabling streaming mode allows lower latency and higher bandwidth of USB transfers but requires lower system memory latency.

Long system memory latency might cause TX buffer underrun/RX buffer overrun when streaming mode is enabled. Streaming mode should be kept enabled when USB controller is initialized and streaming mode should only be disabled if it is determined the system latency does cause the buffer overrun/underrun issues.

19.7.2 USB2 Programming Guidelines

The USB2 controller supports a UHSIC interface. All of these interfaces are directly controlled through USB2 register space.

19.7.2.1 Clock Initialization

After power-on-reset, the USB clocks are disabled. Software can enable the USB clocks by setting CLK_ENB_USBD in CLK_OUT_ENB_L register to ENABLE.

Also, USB stays in reset. So software should bring it out of reset by first setting the SWR_USBD_RST in RST_DEVICES_L register to ENABLE and then setting it to DISABLE.

After the clocks for USB2 are up, software can program any register for USB2 required for proper configuration.

PLLU outputs a 480 MHz clock (FO_UHSIC) for the UHSIC interface. If using any of these interfaces, PLLU should be programmed appropriately.

Any time after USB2 is reset, or the UHSIC PHY comes out of suspend, software needs to wait until 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; otherwise, they are not. There are interrupts associated with this bit: if USB_PHY_CLK_VALID_INT_ENB is set to ENABLE, USB_PHY_CLK_VALID_INT_STS will be set to SET whenever the PHY clock becomes valid. Software can write a 1 to USB_PHY_CLK_VALID_INT_STS to clear the interrupt status.

Before doing anything useful on USB2, one of its interfaces needs to be selected and configured.

19.7.2.2 Selection of UTMIP2 Interface

The UTMIP1 I/O pads are in power-down state at power-on. Bring them out of power-down mode by writing 0 to the following fields (unless writing 1 is indicated):

1. FORCE_PD_POWERDOWN, FORCE_PD2_POWERDOWN, and FORCE_PDZI_POWERDOWN fields in the UTMIP_XCVR_CFG0 register.
2. OTGPD and BIASPD fields in the UTMIP_BIAS_CFG0 register.
3. FORCE_PDDISC_POWERDOWEN, FORCE_PDCHRP_POWERDOWN, and FORCE_PDDR_POWERDOWN fields in the UTMIP_XCVR_CFG1 register.
4. Hold UTMIP2 PHY in reset by writing a 1 to the UTMIP_RESET bit in the IF_USB_SUSP_CTRL register.
5. Program the PLLU parameters to enable the UTMIP2 PLLU clock.
6. Enable UTMIP2 interface by setting UTMIP_PHY_ENB in IF_USB_SUSP_CTRL register to 1.
7. Program the configuration parameters for UTMIP1.
8. Release reset to UTMIP2 by writing 0 to the UTMIP_RESET field in the IF_USB_SUSP_CTRL register.
9. Wait until the PHY clock comes up by checking the USB_PHY_CLK_VALID bit in the IF_USB_SUSP_CTRL register. An interrupt can be generated as explained above.
10. Set the CM field in the USB2D_USBMODE register to HOST mode.
11. Program the USB2 controller to use UTMIP2 PHY by setting the PTS field in the CONTROLLER_2_USB2D_HOSTPC1_DEVLC register to UTMIP (2’b00).

Selection of UHSIC Interface

1. UHSIC I/O pad is in power-down state at power-on. Bring it out of power-down mode by setting the PD_BG, PD_TX, PD_RX, PD_ZI and RPD_DATA, RPD_STROBE fields in the UHSIC_PADS_CFG1 register to 0.
2. Bring the tracking circuit out of power-down mode by clearing the PD_TRK bit.
3. Add 25 µs delay to allow calibration to complete.
4. Tracking circuit can be powered down now by setting PD_TRK=1.
5. Hold UHSIC in reset by writing 1 to the UHSIC_RESET field in the IF_USB_SUSP_CTRL register.
6. Program the PLLU parameters to enable the UHSIC PLLU clock.
7. Select the UHSIC interface by writing 1 to the UHSIC_PHY_ENB field in the IF_USB_SUSP_CTRL register.
8. Program the configuration parameters for UHSIC.

9. Release reset to the UHSIC by writing 0 to the UHSIC_RESET field in the IF_USB_SUSP_CTRL register.
10. Set the CM field in the USB2D_USBMODE register to HOST mode.
11. Program the USB2 controller to use the UHSIC PHY by writing 0 to the PTS field in the CONTROLLER_1_USB2D_HOSTPC1_DEVLC register.
12. Program the UHSIC_HS_POSTAMBLE_OUTPUT_ENABLE field in the UHSIC_TX_CFG0 register to “1”.
13. Wait until the PHY clock comes up by checking the USB_PHY_CLK_VALID bit in the IF_USB_SUSP_CTRL register. An interrupt can be generated as explained above.

19.7.2.3 PHY Clock Control

The PHY clock can be turned off using the PHCD bit in the USB_PORTSC1 register. Note: This bit does not need to be pulsed.

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 the suspend mode, the PHY clock is turned off and USB_PHY_CLK_VALID in the USB2_IF_USB_SUSP_CTRL register is set to 0. Software should make sure that the PHY clock shuts down by polling this bit whenever it places the PHY into suspend mode.

The PHY clock can be turned on automatically on the events mentioned below. Set only the required wakeup enable bits based on the state of the USB as mentioned below. Do not set unnecessary wakeup enable bits. When the PHY clock wakes up under any such event, make sure that the wakeup enable bits are turned off as soon as possible.

To prevent the PHY clock from waking up on unnecessary glitches on the USB pins, software can set the USB_WAKEUP_DEBOUNCE_COUNT field in the USB_SUSP_CTRL register to a non-zero value (between 1-7). This will allow the wakeup event to be debounced by the equivalent number of HCLK cycles.

An interrupt is generated whenever the PHY clock is turned on, which can be checked by checking the value of USB_PHY_CLK_VALID in the USB_SUSP_CTRL register to be 1. The interrupt can be enabled/disabled by setting the USB_PHY_CLK_VALID_INT_ENB bit in the USB_SUSP_CTRL register to 1/0.

To clear the USB_WAKEUP_INT_STS and USB_PHY_CLK_VALID_INT_STS interrupts, software can write a 1 to corresponding bits.

Generally, whenever the PHY is woken up from the suspend mode, first a wakeup event is generated (USB_WAKEUP_INT_STS = 1) followed by a PHY clock valid interrupt (USB_PHY_CLK_VALID_INT_STS = 1) when PHY clock starts up.

19.7.2.4 Bus Signaling Procedure Changes for the UHSIC Interface

To support the UHSIC interface, some changes are required in the bus signaling procedure for USB2 controller (connect and bus reset sequences) as described in the subsections below:

Host Mode - Bus Connect Sequence

The normal connect sequence assumes that port power has been enabled on a downstream port and the device has signaled a “Connect” on the bus. This will generate a port change interrupt and the software takes action from here. Standard EHCI connect interrupt can be used to detect the device connect.

Host Mode: Bus Reset Sequence

This is a reconnect procedure without physically detaching the USB cable. This operation is demanded by software whenever 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 (the PR bit in the 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 the USB 2.0 specification 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 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 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 K1 HSIC because native HSIC support is built-in.

19.7.3 USB3 Programming Guidelines

The USB3 controller supports ICUSB and UTMIP3 interfaces. All of these interfaces are directly controlled through USB3 register space.

19.7.3.1 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, software can program any registers for USB3 required for proper configuration.

PLLU outputs a 60 MHz clock FO_ICUSB for ICUSB interface and a 12 MHz clock FO_USB for UTMIP3 interface. If any of these interfaces are used, PLLU should be programmed appropriately.

Any time after 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 the USB_PHY_CLK_VALID bit in the USB3_IF_USB_SUSP_CTRL register. If this bit is 1, the PHY clocks are up; otherwise they are not. There are interrupts associated with this bit: if USB_PHY_CLK_VALID_INT_ENB is set to ENABLE, USB_PHY_CLK_VALID_INT_STS will be set to SET whenever PHY clock becomes valid. Software can write a 1 to USB_PHY_CLK_VALID_INT_STS to clear the interrupt status.

Before doing anything useful on USB2, one of its interfaces needs to be selected and configured.

19.7.3.2 Selection of the UTMIP3 Interface

1. UTMIP3 I/O pads are in power-down state at power-on. Bring them out of power-down mode by writing the following fields to 0:

- 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_POWERDOWN , FORCE_PDCHRP_POWERDOWN, FORCE_PDDR_POWERDOWN fields in UTMIP_XCVR_CFG1 register.

2. Hold UTMIP3 PHY in reset by writing the UTMIP_RESET bit in the IF_USB_SUSP_CTRL register to 1.
3. Program the PLLU parameters to enable UTMIP3 PLLU clock.
4. Enable the UTMIP3 interface by setting UTMIP_PHY_ENB in the IF_USB_SUSP_CTRL register to 1.
5. Program the configuration parameters for UTMIP3.
6. Release the reset to UTMIP3 by writing 0 to the UTMIP_RESET field in the IF_USB_SUSP_CTRL register.
7. Wait until the PHY clock comes up by checking the USB_PHY_CLK_VALID bit in the IF_USB_SUSP_CTRL register. An interrupt can be generated as explained above.
8. Set the CM field in the USB2D_USBMODE register to HOST mode.
9. Program the USB3 controller to use UTMIP3 PHY by setting the PTS field in CONTROLLER_2_USB2D_HOSTPC1_DEVLC register to UTMIP (2'b00).

19.7.3.3 Selection of UHSIC Interface

1. UHSIC I/O pad is in power-down state at power-on. Bring it out of power-down mode by setting the PD_BG, PD_TX, PD_RX, PD_ZI, RPD_DATA, and RPD_STROBE fields in the UHSIC_PADS_CFG1 register to 0.
2. Bring the tracking circuit out of power-down mode by clearing the PD_TRK bit.
3. Add 25 µs delay to allow calibration to complete.
4. The tracking circuit can be powered down now by setting PD_TRK=1.
5. Hold UHSIC in reset by writing 1 to the UHSIC_RESET field in the IF_USB_SUSP_CTRL register.
6. Program the PLLU parameters to enable the UHSIC PLLU clock.
7. Select the UHSIC interface by writing 1 to the UHSIC_PHY_ENB field in the IF_USB_SUSP_CTRL register.
8. Program the configuration parameters for UHSIC.
9. Release reset to the UHSIC by writing 0 to the UHSIC_RESET field in the IF_USB_SUSP_CTRL register.
10. Set the CM field in the USB2D_USBMODE register to HOST mode.
11. Program the USB3 controller to use the UHSIC PHY by writing 0 to the PTS field in the CONTROLLER_1_USB2D_HOSTPC1_DEVLC register.
12. Program the UHSIC_HS_POSTAMBLE_OUTPUT_ENABLE field in the UHSIC_TX_CFG0 register to "1".
13. Wait until the PHY clock comes up by checking the USB_PHY_CLK_VALID bit in the IF_USB_SUSP_CTRL register. An interrupt can be generated as explained above.

19.7.3.4 Detection of USB Cable Insertion

The USB cable insertion event can be detected by a change in the ID or VBUS pin on the micro-AB connector. Initially, when the cable is not attached, the ID pin on the micro-AB connector is floating and is detected as 1 by Tegra K1 devices because of the weak pull-up on this pin. This makes a Tegra K1 device start up as a B-device, even though a cable is not connected (when the ID pin is seen as a 1 by a Tegra K1 device, there can be no cable or micro-B connected).

When a user attaches the USB cable, one of the following two events occur:

- User connects micro-A end of the cable to the Tegra K1 device. This changes the ID pin to 0. Hence the software can detect the ID change and go to A-device mode. Software can then supply power to the USB and place the Tegra K1 device in host mode.
- User connects micro-B (or mini-B) end of the cable to the Tegra K1 device. The other end of the cable can be either micro-A or standard-A and will be connected to a standard host, respectively.

For both cases, the Host will start driving power on VBUS and software can detect the cable insertion event by checking the VBUS status on corresponding GPIO. Once this is seen, it can place the Tegra K1 device in the device mode.

Note: Both ID and VBUS changes can be detected by enabling the corresponding GPIO interrupts.
USB controller/PHY clocks do not need to be turned on for this detection as this is done using GPIOs.
To detect these events during DPD/LP0 modes, use VBUS/ID wakeup events in the PMC.

19.7.3.5 PHY Clock Control

The PHY clock can be turned off by writing 1 to the PHCD bit in the USB2D_PORTSC1 register. Note: There is no need to pulse this bit.

To turn on the PHY clock, software should write to SUSP_CLR in the USB3_IF_USB_SUSP_CTRL register. This bit must be pulsed by first writing a 1 and then writing a 0 to it.

The current suspend status of the PHY and thereby whether the PHY clock is running can be checked by reading the bit USB_PHY_CLK_VALID in USB_SUSP_CTRL register. If the PHY is placed in suspend, then this bit will be set to 0, else it will be set to 1. Software should make sure that PHY clock shuts down by polling this bit whenever it places the PHY into suspend.

The PHY clock can be turned on automatically on the events mentioned below. Set only the required wakeup enable bits based on the state of the USB as mentioned below. Do not set unnecessary wakeup enable bits. When the PHY clock wakes up under any such event, make sure that the wakeup enable bits are turned off as soon as possible.

To prevent the PHY clock from waking up on unnecessary glitches on the USB pins, software can set the USB_WAKEUP_DEBOUNCE_COUNT field in USB_SUSP_CTRL register to a non-zero value (between 1-7). This will allow the wakeup event to be debounced by the equivalent number of HCLK cycles.

An interrupt is generated whenever the PHY is woken up from suspend, which can be verified by checking that the value of USB_WAKEUP_INT_STS in USB_SUSP_CTRL register is 1. The interrupt can be enabled/disabled by setting the bit USB_WAKEUP_INT_ENB in USB_SUSP_CTRL to 1/0.

An interrupt is generated whenever the PHY clock is turned on, which can be verified by checking that the value of USB_PHY_CLK_VALID_INT_STS in the USB_SUSP_CTRL register is 1. The interrupt can be enabled/disabled by setting the USB_PHY_CLK_VALID_INT_ENB bit in the USB_SUSP_CTRL register to 1/0.

To clear the USB_WAKEUP_INT_STS and USB_PHY_CLK_VALID_INT_STS interrupts, software can write a 1 to the corresponding bits.

Generally, whenever PHY is woken up from suspend, first a wakeup event is generated (USB_WAKEUP_INT_STS = 1) followed by a PHY clock valid interrupt (USB_PHY_CLK_VALID_INT_STS = 1) when the PHY clock starts up.

Note: The PLLU_ENABLE bit in PLLU_BASE register should be always set to ENABLE. All the parameters for PLLU in PLLU_BASE and PLLU_MISC register should be set according to the oscillator frequency used. Not doing that will put the USB PHY PLL in a free-running mode, and can result in undesirable side-effects.
When the USB PHY is suspended, the USB3 clock should not be stopped. It can be reduced to 32 KHz, however. If the system clock is stopped, then it may not be possible to bring up the system on a wakeup event described below.

When USB3 is in suspend, AHB clocks to USB3 controller are turned off, and hence there should be no register access to USB2_CONTROLLER_2_* registers. All registers marked as ARUSB3_IF_* are accessible, and can be used to resume the UTMIP3 PHY clocks.

When a device is not connected, software can set the WKCN bit in USB2D_PORTSC1 register. Make sure that VBUS is also turned on. After this, it can stop the PHY clock by using the method described above. The PHY clock will resume automatically when a device is connected. The method described above can be used to interrupt the processor when the PHY clock is turned on.

With a device connected, software needs to put the USB system under suspend by setting the bit SUSP in USB2D_PORTSC1 register. It needs to wait until this bit is set to 1 by the controller to make sure that the USB system actually went into suspend. It can enable the bits USB_WAKE_ON_RESUME_EN in USB_SUSP_CTRL register and WK_DS in USB2D_PORTSC1 register. Then it can stop the PHY clock as described above. The PHY clock will be turned on automatically if the USB device disconnects or it does a remote wakeup signaling. There will be an interrupt associated with this as described above.

If the software wants to wake up the USB system, it needs to turn on the PHY clock as described above.

19.7.3.6 USB3 Controller Power Gating

Initialization and Shutdown Procedure for USB

1. Bring up USB3 clocks by writing 1 to CLK_ENB_USB3 in ARCLK_RST_CLK_OUT_ENB_H register.
2. Set and clear the SWR_USB3_RST bit in the RST_DEVICES_H register to bring the USB block out of reset.
3. Set the UTMIP_RESET bit in the USB_SUSP_CTRL register to 1 to keep UTMIP3 PHY in reset.
4. Program the UTMIP and PLLU parameters.
   - Set the UTMIP_SUSPEND_EXIT_ON_EDGE bit (bit 22) in the UTMIP_MISC_CFG0 register to 0.
   - Set PLLU_ENABLE to 1, and never deassert it.
5. Set the UTMIP_RESET bit in the 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 the USB_PHY_VBUS_SENSORS register to 1.
   - Enable ID detection interrupt by setting ID_INT_EN in the USB_PHY_VBUS_WAKEUP_ID register.
9. When (VBUS=1) or (ID=0) is detected, do the following:

- Bring the USB3 controller and PHY out of suspend mode by pulsing the 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 point on, the driver can run normal operations.
    - UTMIP should only be suspended by writing 1 to PHCD in the USB2D_PORTSC1 register when the USB cable is connected and only after the USB bus is suspended, for both device and host modes.
11. If a cable disconnect is detected (VBUS = 0) or (ID = 1), then go back to step 4.

19.7.4 UTMIP Programming Guidelines

A Tegra K1 device has 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. Internally, these ports correspond to USB controllers 1, 2, and 3, where controller 1 is often implicitly numbered. Externally, these are USB ports 0, 1, and 2. All UTMI+ PHY interfaces are identical instances of the same design. All three controllers use latest edition of USB core controller.

19.7.4.1 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.

19.7.4.2 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 (UTMIP, UHSIC). Table 66 describes the parameter settings that are dependent on the crystal clock reference frequency. The parameters are specified in decimal notation.

Table 66: 480 MHz PLL_U Output Frequency Fo = (Fi·N)÷(M· 2) With VCO_FREQ=0

Fi (MHz)	12.0MHz	13.0MHz	16.8MHz	19.2MHz	26.0MHz	38.4 MHz	48.0MHz
N (PLLU_DIVN)	960	960	400	200	960	200	960
M (PLLU_DIVM)	12	13	7	4	26	4	12

These parameters can be found in the PLLU_BASE register. The PLLU_OVERRIDE bit should be set to 0. PLLU_VCO_FREQ parameter must also be set to 0.

Note: PLL_U must be properly configured in the Boot ROM. DO NOT change the parameters of PLL_U while the unit is running. The same applies to the USB_PHY_PLL.

19.7.4.3 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 because it cannot do so rapidly enough to meet the protocol. The following table lists the start times, which must be set up in the Boot ROM.

Table 67: PLL Automated Start Times (Must be Set Up in the Boot ROM)

Crystal Frequency	12.0 MHz	13.0 MHz	16.8 MHz	19.2 MHz	26.0 MHz	38.4 MHz	48.0 MHz
PLLU_ENABLE_DELAY_COUNT	2	2	3	3	4	5	6
PLLU_STABLE_COUNT	47	51	66	75	102	150	188
PLL_ACTIVE_DLY_COUNT	8	9	11	12	9	24	31
XTAL_FREQ_COUNT	118	127	165	188	254	375	469

PLLU_ENABLE_DLY_COUNT should be set for at least 1 microsecond, PLLU_STABLE_COUNT should be set for at least 1 ms, the PLL_ACTIVE_DLY_COUNT should be at least 10 ms, and the XTAL_FREQ_COUNT set for about 2.5 ms.

19.7.4.4 Fuse Programming

The Tegra K1 device incorporates fuses to account for process variation in high speed data eye swing. The high speed data eye is controlled through the SETUP[6:0] transceiver pad parameter of the USB control registers. To set this parameter via the fuses, set the UTMIP_SPARE_CFG0[3] bit on all USB ports. This must be done in the Boot ROM. The default is to maintain backward compatibility, which implies no fuses are used by default. The objective is to get the data eye swing as close as possible to 400mV without falling below the mark under typical operating conditions.

19.7.4.5 Programming the Tracking Length Time

The PD_TRK signal is used to power down the bias cell. After 25 microseconds of bias cell operation, the PD_TRK signal can be turned high to save power. This can be automated by programming a timing interval as given in the following table.

Table 68: 25 µs Timer on Bias Cell Tracking (for Set Up in the Boot ROM)

Crystal Frequency	12.0 MHz	13.0 MHz	16.8 MHZ	19.2 MHz	26.0 MHz	38.4 MHz	48.0 MHz
UTMIP_BIAS_PAD_TRK_COUNT (UTMIP_BIAS_CFG1[7:3])	5	6	7	8	11	15	19

All values in the table are decimal. This parameter must be set in the Boot ROM while the crystal clock is stopped. To stop the crystal clock, disable the UTMIP_PHY_XTAL_CLOCKEN bit of register UTMIP_MISC_CFG1 and then re-enable it when done.

Because there is more than one UTMIP sharing the same bias cell, power down is done through a daisy chain, requiring power down across all ports.

19.7.4.6 Powering Down a USB Port Outside of Deep Power Down (DPD)

When a port is known to be disabled, there needs to be a mechanism to stop its power consumption for situations outside of DPD. This section addresses disabling a port while powered up.

Powering down of USB at times other than deep power down should be done by setting all specialized PLL and pad power down pins instead of using the global E_DPD pins of the USB analog components. It is safer to power down the cells through the traditional power down pins when the 3.3V and 1.2V supplies might still be on. This is achieved by setting the following pad controls from UTMIP register space. Before setting the power down bits; save previous registers values so that they may be restored. This type of power down should not be done in the Boot ROM.

Table 69: 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

Pad Control	Analog Cell	Register Bit Settings
PD_CHRP	Transceiver	UTMIP_FORCE_PDCHRP_POWER_DOWN=1
PD_DISC	Transceiver	UTMIP_FORCE_PDDISC_POWER_DOWN=1
PD_CHG	Transceiver	UTMIP_PD_CHG=1
PD	Bias	UTMIP_BIASPD=1
PD_TRK	Bias	UTMIP_FORCE_PDTRK_POWER_DOWN=1
OTG_PD	Bias	UTMIP_OTGPD=1
ID_PD	Bias	UTMIP_IDPD_SEL=1, UTMIP_IDPD_VAL=1
VBUS_WAKEUP_PD	Bias	UTMIP_VBUS_WAKEUP_POWER_DOWN=1
ENABLE	PHY PLL	UTMIP_FORCE_PLL_ENABLE_POWERDOWN=1
ACTIVE	PHY PLL	UTMIP_FORCE_PLL_ACTIVE_POWERDOWN=1
PD_SAMP_A/C	PHY_PLL	UTMIP_FORCE_PD_SAMP_A/C_POWERDOWN=1
ENABLE	PLL U	UTMIP_FORCE_PLLU_POWERDOWN=1
UTMIP	PHY	UTMIP_PHY_XTAL_CLOCKEN=0

19.7.4.7 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.

In the Boot ROM, the A debounce period should be set to 10 ms. This is dependent on the crystal frequency scale. The following table shows how it is programmed.

Table 70: 10 ms Timer on the A Debounce Period

Crystal Frequency	12.0 MHz	13.0 MHz	16.8 MHZ	19.2 MHz	26.0 MHz	38.4 MHz	48.0 MHz
UTMIP_BIAS_DEBOUNCE_A	30000	32500	42000	48000	65000	96000	120000

To program values of 96000 and 120000 for crystal clock frequencies of 38.4 and 48 MHz, respectively, these steps must be followed:

1. In the debounce register, program 48000 for a 38.4 MHz crystal clock or 60000 for a 60 MHz clock.
2. Set UTMIP_BIAS_DEBOUNCE_TIMESCALE to 1 by writing into the USB2_UTMIP_BIAS_CFG1[1] register field.

19.7.4.8 Deep Power Down Behavior

Tegra K1 devices' deep power down behavior for USB wake-up events is controlled in the PMC via registers USB_DEBOUNCE_DLY, USB_AO and the WAKE_MASK (USB_EVENT field) of the PMC unit. These control the wake-up events, their debounce duration, and their debounce length while powered down. USB port 0 has a possible event on ID or VBUS detection. These can be configured at startup time. These registers take over from the UTMIP registers when the chip is powered down. The programming depends on the context of the chip. For example, if USB port 0 is a dedicated host port then the VBUS wake-up event should be kept powered down. Conversely, a dedicated device mode port may not want to wake-up on the detection of an ID connector and can chose to power down ID.

19.7.4.9 Miscellaneous Boot ROM Fields

The UTMIP fields in the following table must be programmed at Boot ROM time. These should be set when the USB device is held in reset.

Field	Value
UTMIP_FS_PREAMBLE_J (UTMIP_TX_CFG0)	1
UTMIP_PD_CHG (UTMIP_BAT_CHRG_CFG0)	1
UTMIP_XCVR_LSBIAS_SEL (UTMIP_XCVR_CFG0)	0
UTMIP_SPARE_CFG0[3]	1
UTMIP_IDLE_WAIT (UTMIP_HSRX_CFG0)	17
UTMIP_ELASTIC_LIMIT (UTMIP_HSRX_CFG0)	16
UTMIP_HS_SYNC_START_DLY (UTMIP_HSRX_CFG1)	9

19.7.4.10 PHY Resets

Additional controls have been provided to stop the USB PHY (and its clocks) by issuing a reset to the PHY only (as opposed to a reset encompassing the controller and the PHY). This is akin to the Reset found in the UTMI+ standard. This is done in both controllers. See the UTMIP_RESET field of the USB_SUSP_CTRL register for more information.

19.7.4.11 Static Boot ROM Configuration for UTMIP

The boot ROM configuration for UTMIP should be done while the unit is held in Reset. For correct behavior, the following sequence MUST be followed:

1. Apply Reset to the USB controller (and UTMIP).
2. Run the crystal clock for approximately 5 microseconds while the UTMIP is in reset.
3. Stop the crystal clock by setting UTMIP_PHY_XTAL_CLOCKEN Low. This only stops the crystal clocks in the UTMIP units.
4. Program PLL_U as described in the “PLL_U Programming” section.
5. Program automatic PLL start times as described in the “PLL_U and USB_PHY_PLL Automatic Startup Times” section.
6. Remove power downs from USB_PHY_PLL ACTIVE/ENABLE/PD_SAMP_A/C and PLL_U.
7. Program the tracking duration as described in “Programming the Tracking Length Time.” Remove the power downs from PD (Bias) and disable PD_TRK. After 25 µs (tracking time), enable PD_TRK by writing 0 into UTMIP_FORCE_PDTRK_POWER_DOWN.
8. Once PD (Bias) is disabled, disable OTG_PD after 1 µs.
9. Remove powerdowns from ID_PD and VBUS_WAKEUP_PD. This can be removed earlier, as there is no timing or sequential relation with other signals.
10. Once the PLL_U, USB_PHY_PLL, and Bias pad are ready, remove PD (Transceiver). After 400 ns, the USB pad is ready for HS/FS/LS operation.
11. There are other powerdowns which can be removed if needed (as such not needed for Boot ROM). These are PD / PD_ZI / PD_DR / PD_CHRP / PD_DISC / PD_CHG.
12. Program the debouncer length time as described in ”Extending Debouncer Period Length.”
13. Program various static parameters of the UTMIP as described in “Miscellaneous Boot ROM Fields.”
14. Restart the crystal clock by setting UTMIP_PHY_XTAL_CLOCKEN.

15. Resume any previous USB programming work that falls outside of UTMIP and release reset.From the UTMIP perspective it does not matter when reset is released, as long as it is after step 8. It will take about 3.5 ms before the 60 MHz clock appears once reset is released.

From a UTMIP perspective, ensure that all the cabled USB port PHYs are configured similarly in the Boot ROM.

19.7.4.12 EHCI Deviations

- Embedded Transaction Translator – Allows direct attachment of FS and LS devices in host mode without the need for a companion controller.
- Embedded design interface – This core does not have a PCI Interface and therefore the PCI configuration registers described in the EHCI specification are not applicable.

Programmable Physical Interface Behavior

This design supports multiple Physical interfaces which can operate in differing modes when the core is configured with software programmable Physical Interface Modes. Software programmability allows the selection of the Physical interface part during the board design phase instead of during the chip design phase. The control bits for selecting the Physical Interface operating mode have been added to the HOSTPCx register providing a capability that is not defined by EHCI.

Port Reset

The port connect methods specified by EHCI require setting the port reset bit in the PORTSCx register for a minimum duration of 10ms. Due to the complexity required to support the attachment of devices that are not high speed, there are counter already present in the design that can count the 10ms reset pulse to alleviate the requirement of the software to measure this duration. The basic connection is then summarized as the following:

- [Port Change Interrupt] Port connect change occurs to notify the host controller driver that a device has attached.
- Software shall write a ‘1’ to the reset the device.
- Software shall write a ‘0’ to the reset the device after 10 ms.
- Driver needs to wait until port change interrupt is asserted and port reset is cleared

> **Note:** Should the EHCI host controller driver attempt to write a ‘0’ to the reset bit while a reset is in progress the write will be ignored and the reset will continue until completion.

- [Port Change Interrupt] Port enable change occurs to notify the host controller that the device is now operational and at this point the port speed has been determined.

Port Speed Detection

After the port change interrupt indicates that a port is enabled, the EHCI stack should determine the port speed. Unlike the EHCI implementation which will re-assign the port owner for any device that does not connect at High-Speed, this host controller supports direct attach of non High-Speed devices. Therefore, the following differences are important regarding port speed detection:

- Port Owner is read-only and always reads 0.
- A 2-bit Port Speed indicator (PSPD) has been added to HOSTPCx to provide the current operating speed of the port to the host controller driver.

Port Reset duration is 55 ms instead of 50 ms.

Port Suspend (PORTSC.Suspend) bit is set immediately after setting the bit. As per EHCI a 4 ms delay is expected for this bit to be set.

Tegra K1 devices also have the following cases that are not directly compliant to EHCI:

- USB controller/PHY interface initialization
- PHY Clock shutdown/wakeup procedure during suspend/resume/connect/disconnect
- PMC logic use for special low power modes during suspend/LP0

19.7.5 Cold Boot

Step 1

The Boot ROM enables USB2.0 related PLLs with software override.

The Boot ROM deasserts reset to the XUSB PADCTL block.

Step 2

The PAD driver switches USB 2.0 related PLLs to under hardware control after the PAD driver is loaded.

- Program the following CAR registers to set up hardware control of UTMIPLL:
  - CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_USE_LOCKDET] to ‘1’
  - CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_CLK_ENABLE_SWCTL] to ‘0’
  - CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_SEQ_START_STATE] to ‘1’
  - CLK_RST_CONTROLLER_UTMIP_PLL_CFG1_0[UTMIP_FORCE_PLL_ENABLE_POWERUP] to ‘0’
  - CLK_RST_CONTROLLER_UTMIP_PLL_CFG1_0[UTMIP_FORCE_PLL_ENABLE_POWERDOWN] to ‘0’
- Program the following CAR registers to set up software override of UTMIPLL’s IDDQ control.
  - CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_IDDQ_SWCTL] to ‘1’
  - CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_IDDQ_OVERRIDE_VALUE] to ‘0’
- Program the following CAR register to enable hardware control to UTMIPLL:
  - Wait 1 µs then set CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_SEQ_ENABLE] to ‘1’
- Program the following CAR registers to set up hardware control of PLLU:
  - CLK_RST_CONTROLLER_PLLU_HW_PWRDN_CFG0_0[PLLU_USE_LOCKDET] to ‘1’
  - CLK_RST_CONTROLLER_PLLU_HW_PWRDN_CFG0_0[PLLU_CLK_ENABLE_SWCTL] to ‘0’
  - CLK_RST_CONTROLLER_PLLU_HW_PWRDN_CFG0_0[PLLU_CLK_SWITCH_SWCTL] to ‘0’
  - CLK_RST_CONTROLLER_PLLU_HW_PWRDN_CFG0_0[PLLU_SEQ_START_STATE] to ‘1’
- Program the following CAR register to set up hardware control of PLLU when all non-HSIC USB2.0 ports are assigned to XUSB:
  - CLK_RST_CONTROLLER_PLLU_HW_PWRDN_CFG0_0[PLLU_SEQ_IN_SWCTL] to ‘0’
- Program the following CAR register to enable hardware control of PLLU:
  - Wait 1 µs then set CLK_RST_CONTROLLER_PLLU_HW_PWRDN_CFG0_0[PLLU_SEQ_ENABLE] to ‘1’
- Program the following CAR register bits to disable the override to PLLU:
  - CLK_RST_CONTROLLER_UTMIP_PLL_CFG1_0[UTMIP_FORCE_PLLU_POWERUP] to ‘0’

- CLK_RST_CONTROLLER_UTMIP_PLL_CFG1_0[UTMIP_FORCE_PLLU_POWERDOWN] to ‘0’
- CLK_RST_CONTROLLER_PLLU_BASE_0[PLLU_OVERRIDE] to ‘0’

The PAD driver enables USB3.0 related PLLs.

- Program the following CAR register bits to ‘0’ to move the REFPLLE and PLLE out of IDDQ:
  - CLK_RST_CONTROLLER_PLLREFE_MISC_0[PLLREFE_IDDQ]
  - CLK_RST_CONTROLLER_PLLE_MISC_0[PLLE_IDDQ_OVERRIDE_VALUE]
- Program the following CAR register bits to configure the REFPLLE:
  - CLK_RST_CONTROLLER_PLLREFE_BASE_0[PLLREFE_MDIV]
  - CLK_RST_CONTROLLER_PLLREFE_BASE_0[PLLREFE_NDIV]
  - CLK_RST_CONTROLLER_PLLREFE_BASE_0[PLLREFE_PLDIV]
  - CLK_RST_CONTROLLER_PLLREFE_BASE_0[PLLREFE_KVCO]
  - CLK_RST_CONTROLLER_PLLREFE_BASE_0[PLLREFE_KCP]
- Set the following CAR register bit to ‘1’ to enable the PLL:
  - CLK_RST_CONTROLLER_PLLREFE_BASE_0[PLLREFE_ENABLE]
- Program the following CAR register to enable REFPLLE LOCK output
  - CLK_RST_CONTROLLER_PLLREFE_MISC_0[PLLREFE_LOCK_ENABLE] to ‘1’
- Poll the following CAR register to ensure REFPLLE is locked:
  - CLK_RST_CONTROLLER_PLLREFE_MISC_0[PLLREFE_LOCK] equals ‘1’
- Program the following CAR register bits to configure the PLLE according to the frequency of osc_clk:
  - CLK_RST_CONTROLLER_PLLE_BASE_0[PLLE_MDIV]
  - CLK_RST_CONTROLLER_PLLE_BASE_0[PLLE_NDIV]
  - CLK_RST_CONTROLLER_PLLE_BASE_0[PLLE_FDIV4B]
  - CLK_RST_CONTROLLER_PLLE_BASE_0[PLLE_PLDIV_CML]
- Set the following CAR register bit to ‘1’ to enable the PLL:
  - CLK_RST_CONTROLLER_PLLE_BASE_0[PLLE_ENABLE]
- Program the following CAR register to enable PLL LOCK output:
  - CLK_RST_CONTROLLER_PLLE_MISC_0[PLLE_LOCK_ENABLE] to 1’b1
- Poll the following CAR register to ensure PLLE is locked:
  - CLK_RST_CONTROLLER_PLLE_MISC_0[PLLE_LOCK] equals ‘1’

The PAD driver switches USB 3.0 related PLLs to under hardware control.

- Program the following CAR registers to set up hardware control of PLLE
  - CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_USE_LOCKDET] to ‘1’
  - CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_ENABLE_SWCTL] to ‘0’
  - CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_SS_SWCTL] to ‘0’
  - CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_SEQ_START_STATE] to ‘1’
  - CLK_RST_CONTROLLER_PLLE_MISC_0[PLLE_IDDQ_SWCTL] to ‘0’

- Program the following CAR register to enable hardware control of PLLE:
  - Wait 1 µs then set CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_SEQ_ENABLE] to '1'
- Program the following CAR registers to set up hardware control of XUSB Brick's PLL, where XUSBIO_PLL is used by either or both XUSB and PCIE and SATA_PLL is used by either XUSB or SATA:
  - CLK_RST_CONTROLLER_XUSBIO_PLL_CFG0_0[XUSBIO_PADPLL_USE_LOCKDET] to '1'
  - CLK_RST_CONTROLLER_XUSBIO_PLL_CFG0_0[XUSBIO_CLK_ENABLE_SWCTL] to '0'
  - CLK_RST_CONTROLLER_XUSBIO_PLL_CFG0_0[XUSBIO_PADPLL_RESET_SWCTL] to '0'
  - CLK_RST_CONTROLLER_XUSBIO_PLL_CFG0_0[XUSBIO_SEQ_START_STATE] to '1'
  - CLK_RST_CONTROLLER_SATA_PLL_CFG0_0[SATA_PADPLL_USE_LOCKDET] to '1'
  - CLK_RST_CONTROLLER_SATA_PLL_CFG0_0[SATA_PADPLL_RESET_SWCTL] to '0'
  - CLK_RST_CONTROLLER_SATA_PLL_CFG0_0[SATA_SEQ_START_STATE] to '1'
- Wait 1 µs
- Program the following CAR registers to enable hardware control of the XUSB Brick's PLL:
  - CLK_RST_CONTROLLER_XUSBIO_PLL_CFG0_0[XUSBIO_SEQ_ENABLE] to '1'
  - CLK_RST_CONTROLLER_SATA_PLL_CFG0_0[SATA_SEQ_ENABLE] to '1'

Step 3

The PAD driver enables platform specific regulators enable power rails to the pads, VBUS, and pull-up voltage to the VBUS control PMIC's EN input.

The PAD driver assigns the USB ports to the controllers, then programs the port capabilities and pad parameters of ports assigned to XUSB after booting to OS.

- Program the following XUSB PADCTL registers to assign the USB2.0 ports to XUSB, according to the platform-specific configuration:
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_HSIC_PAD_PORT1]
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_HSIC_PAD_PORT0]
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT2]
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT1]
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT0]
- Program the following XUSB PADCTL registers to assign the port capabilities for USB2.0 ports owned by XUSB, according to the platform specific configuration:
  - XUSB_PADCTL_USB2_PORT_CAP_0[PORT2_CAP] for host/disabled
  - XUSB_PADCTL_USB2_PORT_CAP_0[PORT2_INTERNAL] for whether its internal port
  - XUSB_PADCTL_USB2_PORT_CAP_0[PORT1_CAP] for host/disabled
  - XUSB_PADCTL_USB2_PORT_CAP_0[PORT1_INTERNAL] for whether its internal port
  - XUSB_PADCTL_USB2_PORT_CAP_0[PORT0_CAP] for host/disabled
  - XUSB_PADCTL_USB2_PORT_CAP_0[PORT0_INTERNAL] for whether its internal port
- Program the following XUSB PADCTL registers to '0x7' to disable the overcurrent signal mapping for USB 2.0 ports owned by XUSB:
  - XUSB_PADCTL_SNPS_OC_MAP_0[CONTROLLER1_OC_PIN]

- XUSB_PADCTL_SNPS_OC_MAP_0[CONTROLLER2_OC_PIN]
- XUSB_PADCTL_SNPS_OC_MAP_0[CONTROLLER3_OC_PIN]
- XUSB_PADCTL_USB2_OC_MAP_0[PORT2_OC_PIN]
- XUSB_PADCTL_USB2_OC_MAP_0[PORT1_OC_PIN]
- XUSB_PADCTL_USB2_OC_MAP_0[PORT0_OC_PIN]
- XUSB_PADCTL_OC_DET_0[VBUS_ENABLE0_OC_MAP]
- XUSB_PADCTL_OC_DET_0[VBUS_ENABLE1_OC_MAP]
- XUSB_PADCTL_OC_DET_0[VBUS_ENABLE1_OC_MAP]

- Program the following XUSB PADCTL registers to assign the SuperSpeed port mapping to USB2.0 ports owned by XUSB, where the SuperSpeed ports inherit their port capabilities from the USB2.0 ports they mapped to, according to the platform specific configuration:
  - XUSB_PADCTL_SS_PORT_MAP_0[PORT1_MAP]
  - XUSB_PADCTL_SS_PORT_MAP_0[PORT0_MAP]
- Program the following XUSB PADCTL registers to assign the IOPHY lanes to XUSB, PCIE, or SATA, according to the platform-specific configuration
  - XUSB_PADCTL_USB3_PAD_MUX_0[PCIE_PAD_LANE0]
  - XUSB_PADCTL_USB3_PAD_MUX_0[PCIE_PAD_LANE1]
  - XUSB_PADCTL_USB3_PAD_MUX_0[PCIE_PAD_LANE2]
  - XUSB_PADCTL_USB3_PAD_MUX_0[PCIE_PAD_LANE3]
  - XUSB_PADCTL_USB3_PAD_MUX_0[PCIE_PAD_LANE4]
  - XUSB_PADCTL_USB3_PAD_MUX_0[SATA_PAD_LANE0]
- Program the following XUSB PADCTL registers to assign the PAD and PLL parameters of ports owned by XUSB, according to the platform-specific configuration:
  - XUSB_PADCTL_IOPHY_PLL_P0_CTL1_0
  - XUSB_PADCTL_IOPHY_PLL_P0_CTL2_0
  - XUSB_PADCTL_IOPHY_PLL_P0_CTL3_0
  - XUSB_PADCTL_IOPHY_PLL_P0_CTL4_0
  - XUSB_PADCTL_IOPHY_PLL_S0_CTL1_0
  - XUSB_PADCTL_IOPHY_PLL_S0_CTL2_0
  - XUSB_PADCTL_IOPHY_PLL_S0_CTL3_0
  - XUSB_PADCTL_IOPHY_PLL_S0_CTL4_0
  - XUSB_PADCTL_IOPHY_USB3_PAD0_CTL_1_0
  - XUSB_PADCTL_IOPHY_USB3_PAD0_CTL_2_0
  - XUSB_PADCTL_IOPHY_USB3_PAD0_CTL_3_0
  - XUSB_PADCTL_IOPHY_USB3_PAD0_CTL_4_0
  - XUSB_PADCTL_IOPHY_USB3_PAD1_CTL_1_0
  - XUSB_PADCTL_IOPHY_USB3_PAD1_CTL_2_0
  - XUSB_PADCTL_IOPHY_USB3_PAD1_CTL_3_0
  - XUSB_PADCTL_IOPHY_USB3_PAD1_CTL_4_0

- XUSB_PADCTL_IOPHY_MISC_PAD_P0_CTL_1_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P0_CTL_2_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P0_CTL_3_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P0_CTL_4_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P0_CTL_5_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P0_CTL_6_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P1_CTL_1_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P1_CTL_2_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P1_CTL_3_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P1_CTL_4_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P1_CTL_5_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P1_CTL_6_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P2_CTL_1_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P2_CTL_2_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P2_CTL_3_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P2_CTL_4_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P2_CTL_5_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P2_CTL_6_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P3_CTL_1_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P3_CTL_2_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P3_CTL_3_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P3_CTL_4_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P3_CTL_5_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P3_CTL_6_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P4_CTL_1_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P4_CTL_2_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P4_CTL_3_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P4_CTL_4_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P4_CTL_5_0
- XUSB_PADCTL_IOPHY_MISC_PAD_P4_CTL_6_0
- XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_1_0
- XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_2_0
- XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_3_0
- XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_4_0
- XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_5_0
- XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_6_0
- XUSB_PADCTL_USB2_OTG_PAD0_CTL_0_0
- XUSB_PADCTL_USB2_OTG_PAD0_CTL_1_0
- XUSB_PADCTL_USB2_OTG_PAD1_CTL_0_0

- XUSB_PADCTL_USB2_OTG_PAD1_CTL_1_0
- XUSB_PADCTL_USB2_OTG_PAD2_CTL_0_0
- XUSB_PADCTL_USB2_OTG_PAD2_CTL_1_0
- XUSB_PADCTL_USB2_BIAS_PAD_CTL_0_0
- XUSB_PADCTL_USB2_BIAS_PAD_CTL_1_0
- XUSB_PADCTL_HSIC_PAD0_CTL_0_0
- XUSB_PADCTL_HSIC_PAD0_CTL_1_0
- XUSB_PADCTL_HSIC_PAD0_CTL_2_0
- XUSB_PADCTL_HSIC_PAD1_CTL_0_0
- XUSB_PADCTL_HSIC_PAD1_CTL_1_0
- XUSB_PADCTL_HSIC_PAD1_CTL_2_0
- XUSB_PADCTL_HSIC_STRB_TRIM_CONTROL_0

- Program the following XUSB PADCTL register bits to ‘0’ to disable power down of the PAD:
  - XUSB_PADCTL_USB2_BATTERY_CHRG_BIASPAD_0[PD_OTG]
  - XUSB_PADCTL_USB2_OTG_PAD0_CTL_0_0[PD]
  - XUSB_PADCTL_USB2_OTG_PAD1_CTL_0_0[PD]
  - XUSB_PADCTL_USB2_OTG_PAD2_CTL_0_0[PD]
  - XUSB_PADCTL_USB2_OTG_PAD0_CTL_0_0[PD2]
  - XUSB_PADCTL_USB2_OTG_PAD1_CTL_0_0[PD2]
  - XUSB_PADCTL_USB2_OTG_PAD2_CTL_0_0[PD2]
  - XUSB_PADCTL_USB2_OTG_PAD0_CTL_0_0[PD_ZI]
  - XUSB_PADCTL_USB2_OTG_PAD1_CTL_0_0[PD_ZI]
  - XUSB_PADCTL_USB2_OTG_PAD2_CTL_0_0[PD_ZI]
  - XUSB_PADCTL_USB2_OTG_PAD0_CTL_1_0[PD_DR]
  - XUSB_PADCTL_USB2_OTG_PAD1_CTL_1_0[PD_DR]
  - XUSB_PADCTL_USB2_OTG_PAD2_CTL_1_0[PD_DR]
  - XUSB_PADCTL_USB2_BIAS_PAD_CTL_0_0[PD]
  - XUSB_PADCTL_USB2_BIAS_PAD_CTL_0_0[PD_TRK]
  - XUSB_PADCTL_HSIC_PAD0_CTL_1_0[PD_TX]
  - XUSB_PADCTL_HSIC_PAD1_CTL_1_0[PD_TX]
  - XUSB_PADCTL_HSIC_PAD0_CTL_1_0[PD_TRX]
  - XUSB_PADCTL_HSIC_PAD1_CTL_1_0[PD_TRX]
  - XUSB_PADCTL_HSIC_PAD0_CTL_1_0[PD_RX]
  - XUSB_PADCTL_HSIC_PAD1_CTL_1_0[PD_RX]
  - XUSB_PADCTL_HSIC_PAD0_CTL_1_0[PD_ZI]
  - XUSB_PADCTL_HSIC_PAD1_CTL_1_0[PD_ZI]
- Program the following XUSB PADCTL register bits to ‘0’ to release the XUSB SS wake logic state latching:
  - XUSB_PADCTL_ELPG_PROGRAM_0[SSP0_ELPG_CLAMP_EN]

- XUSB_PADCTL_ELPG_PROGRAM_0[SSP0_ELPG_CLAMP_EN_EARLY]
- XUSB_PADCTL_ELPG_PROGRAM_0[SSP0_ELPG_VCORE_DOWN]
- XUSB_PADCTL_ELPG_PROGRAM_0[SSP1_ELPG_CLAMP_EN]
- XUSB_PADCTL_ELPG_PROGRAM_0[SSP1_ELPG_CLAMP_EN_EARLY]
- XUSB_PADCTL_ELPG_PROGRAM_0[SSP1_ELPG_VCORE_DOWN]

- Program the following XUSB PADCTL register fields to enable the pad override to put the pad of the lanes assigned to XUSB in low power mode. These pads should be brought out of low power mode during SS clock frequency increasing sequence.
  - XUSB_PADCTL_IOPHY_MISC_PAD_P0_CTL_3_0[RX_IDLE_MODE] to '0'
  - XUSB_PADCTL_IOPHY_MISC_PAD_P0_CTL_3_0[RX_IDLE_MODE_OVRD] to '1'
  - XUSB_PADCTL_IOPHY_MISC_PAD_P1_CTL_3_0[RX_IDLE_MODE] to '0'
  - XUSB_PADCTL_IOPHY_MISC_PAD_P1_CTL_3_0[RX_IDLE_MODE_OVRD] to '1'
  - XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_3_0[RX_IDLE_MODE] to '0'
  - XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_3_0[RX_IDLE_MODE_OVRD] to '1'

Note: To avoid conflict or ambiguity, the XUSB controller should be under reset when updating the port assignments.

Step 4

The PAD driver programs the clocks and deasserts the resets to the controllers

- Set the following CAR register bit to '1' to enable the clocks to XUSB:
  - CLK_RST_CONTROLLER_CLK_ENB_W_SET_0[SET_CLK_ENB_XUSB]
- Set the following CAR register bits to '1' to enable the clocks to individual XUSB partitions:
  - CLK_RST_CONTROLLER_CLK_ENB_U_SET_0[SET_CLK_ENB_XUSB_HOST]
  - CLK_RST_CONTROLLER_CLK_ENB_W_SET_0[SET_CLK_ENB_XUSB_SS]
- Program the following CAR register bits to set the source of XUSB clocks, where PLLP_OUT0 runs at 408 MHz.
  - CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_CORE_HOST_0[XUSB_CORE_HOST_CLK_SRC] to 'PLLP_OUT0'
  - CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_CORE_HOST_0[XUSB_CORE_HOST_CLK_DIVISOR] to '0x6'
  - CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_FALCON_0[XUSB_FALCON_CLK_SRC] to 'PLLP_OUT0'
  - CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_FALCON_0[XUSB_FALCON_CLK_DIVISOR] to '0x2'
  - CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_FS_0[XUSB_FS_CLK_SRC] to 'FO_48M'
  - CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_FS_0[XUSB_FS_CLK_DIVISOR] to '0x0'
  - CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_SS_0[XUSB_SS_CLK_SRC] to 'HSIC_480'
  - CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_SS_0[XUSB_SS_CLK_DIVISOR] to '0x6'
- Set the following CAR register bits to '0' to deassert reset to XUSB:
  - CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_HOST_RST]
  - CLK_RST_CONTROLLER_RST_DEVICES_W_0[SWR_XUSB_SS_RST]

Step 5

The PAD driver brings the IOPHY out of IDDQ.

- Program the following XUSB PADCTL registers to '1' to bring specific lanes of IOPHY out of IDDQ. Only lanes that are used in the platform are required to be bring out of IDDQ:
  - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK0]
  - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK1]
  - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK2]
  - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK3]
  - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK4]
  - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_SATA_PAD_IDDQ_DISABLE_MASK0]
- Alternatively, program the following XUSB PADCTL register to '0' to bring all lanes of IOPHY out of IDDQ:
  - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE]

The PAD driver releases the always on PAD muxing logic state latching.

- Program the following XUSB PADCTL register bits:
- Wait 1 µs
  - XUSB_PADCTL_ELPG_PROGRAM_0[AUX_MUX_LP0_CLAMP_EN] to '0'
- Wait 100 µs
  - XUSB_PADCTL_ELPG_PROGRAM_0[AUX_MUX_LP0_CLAMP_EN_EARLY] to '0'
- Wait 100 µs
  - XUSB_PADCTL_ELPG_PROGRAM_0[AUX_MUX_LP0_VCORE_DOWN] to '0'

The PAD driver enables the VBUS to the USB ports.

- Program the following XUSB PADCTL registers to assign the over current signal mapping for USB 2.0 ports owned by XUSB, according to the platform-specific configuration:
  - XUSB_PADCTL_SNPS_OC_MAP_0[CONTROLLER1_OC_PIN]
  - XUSB_PADCTL_SNPS_OC_MAP_0[CONTROLLER2_OC_PIN]
  - XUSB_PADCTL_SNPS_OC_MAP_0[CONTROLLER3_OC_PIN]
  - XUSB_PADCTL_USB2_OC_MAP_0[PORT2_OC_PIN]
  - XUSB_PADCTL_USB2_OC_MAP_0[PORT1_OC_PIN]
  - XUSB_PADCTL_USB2_OC_MAP_0[PORT0_OC_PIN]
  - XUSB_PADCTL_OC_DET_0[VBUS_ENABLE0_OC_MAP]
  - XUSB_PADCTL_OC_DET_0[VBUS_ENABLE1_OC_MAP]
  - XUSB_PADCTL_OC_DET_0[VBUS_ENABLE2_OC_MAP]
- Write '1' to the following XUSB PADCTL register bits to clear possible false reporting of overcurrent events before the over current signal mappings are properly programmed:
  - XUSB_PADCTL_OC_DET_0[OC_DETECTED0]
  - XUSB_PADCTL_OC_DET_0[OC_DETECTED1]
  - XUSB_PADCTL_OC_DET_0[OC_DETECTED2]

- XUSB_PADCTL_OC_DET_0[OC_DETECTED3]
- XUSB_PADCTL_OC_DET_0[OC_DETECTED_VBUS_PAD0]
- XUSB_PADCTL_OC_DET_0[OC_DETECTED_VBUS_PAD1]
- XUSB_PADCTL_OC_DET_0[OC_DETECTED_VBUS_PAD2]

- Set the following XUSB PADCTL register bits to '1' to enable the VBUS of the host ports
  - XUSB_PADCTL_OC_DET_0[VBUS_ENABLE0]
  - XUSB_PADCTL_OC_DET_0[VBUS_ENABLE1]
  - XUSB_PADCTL_OC_DET_0[VBUS_ENABLE2]

Step 6

The xHCI PEP driver initializes IPFS registers.

- Program the following XUSB IPFS registers to allow software accesses to XUSB's MMIO registers:
  - XUSB_HOST_AXI_BAR0_START_0[AXI_BAR0_START] to '0x70090'
  - XUSB_HOST_AXI_BAR0_SZ_0[AXI_BAR0_SIZE] to '0x00008'
  - XUSB_HOST_FPCI_BAR0_0[FPCI_BAR0_START] to '0x0010000'
  - XUSB_HOST_FPCI_BAR0_0[FPCI_BAR0_ACCESS_TYPE] to '0'
- Program the following XUSB IPFS register to enable the XUSB host:
  - XUSB_HOST_CONFIGURATION_0[EN_FPCI] to '1'

Note: XUSB configuration address mapping is hardcoded in IPFS.

The xHCI PEP driver initializes XUSB's configuration registers.

- Program the following XUSB configuration registers to initialize XUSB:
  - NV_PROJ__XUSB_CFG_1_BUS_MASTER to '1'
  - NV_PROJ__XUSB_CFG_1_MEMORY_SPACE to '1'
  - NV_PROJ__XUSB_CFG_4_BASE_ADDRESS to '0x02000'

Step 6

The xHCI PEP driver loads XUSB firmware.

The xHCI driver initializes the host controller.

19.7.6 LP0

19.7.6.1 LP0 Entry

Step 1

The xHCI driver performs a context save operation as described in section 4.23.2 of the xHCI specification.

The xHCI PEP driver performs an XUSB specific context save operation.

The xHCI PEP driver performs an XUSB IPFS specific context save operation.

- Read and store the value of the following registers
  - XUSB_HOST_MSI_BAR_SZ_0
  - XUSB_HOST_MSI_AXI_BAR_ST_0

- XUSB_HOST_MSI_FPCI_BAR_ST_0
- XUSB_HOST_MSI_VEC0_0
- XUSB_HOST_MSI_EN_VEC0_0
- XUSB_HOST_FPCI_ERROR_MASKS_0
- XUSB_HOST_INTR_MASK_0
- XUSB_HOST_IPFS_INTR_ENABLE_0
- XUSB_HOST_UFPCI_CONFIG_0
- XUSB_HOST_CLKGATE_HYSTERESIS_0
- XUSB_HOST_XUSB_HOST_MCCIF_FIFOCTRL_0

Step 2

The System Power Management driver programs the PMC USB2.0 sleepwalk logic as described in the “PMC Programming” section.

The System Power Management driver should enable the wake events according to the port states and Wake-on-Connect, Wake-on-Disconnect, and Wake-on-Over-Current settings of the XUSB registers ports accordingly.

The System Power Management driver enables wake events from USB ports.

- Set the following PMC register bits to ‘1’ to set the wake signal active level to ‘HIGH’
  - APBDEV_PMC_WAKE2_LVL_0[5] for VBUS wakeup
  - APBDEV_PMC_WAKE2_LVL_0[6] for ID wakeup
  - APBDEV_PMC_WAKE2_LVL_0[7] for USB2.0 port 0 wakeup
  - APBDEV_PMC_WAKE2_LVL_0[8] for USB2.0 port 1 wakeup
  - APBDEV_PMC_WAKE2_LVL_0[9] for USB2.0 port 2 wakeup
  - APBDEV_PMC_WAKE2_LVL_0[10] for HSIC port 0 wakeup
  - APBDEV_PMC_WAKE2_LVL_0[11] for HSIC port 1 wakeup
  - APBDEV_PMC_WAKE2_LVL_0[26] for USB3.0 ports wakeup
- Set the following PMC register bits to ‘1’ to enable USB wake events
  - APBDEV_PMC_WAKE2_MASK_0[5] for VBUS wakeup
  - APBDEV_PMC_WAKE2_MASK_0[6] for ID wakeup
  - APBDEV_PMC_WAKE2_MASK_0[7] for USB2.0 port 0 wakeup
  - APBDEV_PMC_WAKE2_MASK_0[8] for USB2.0 port 1 wakeup
  - APBDEV_PMC_WAKE2_MASK_0[9] for USB2.0 port 2 wakeup
  - APBDEV_PMC_WAKE2_MASK_0[10] for HSIC port 0 wakeup
  - APBDEV_PMC_WAKE2_MASK_0[11] for HSIC port 1 wakeup
  - APBDEV_PMC_WAKE2_MASK_0[26] for USB3.0 ports wakeup

Step 3

The system software puts the system in LP0.

19.7.6.2 LP0 Exit

After exiting LP0, the PMC will restore the partition power gating to the state before LP0 entry. This means if a partition was power gated before entering LP0, that partition would stay power gated after exiting LP0, and only partitions that were powered before entering LP0 would have their power restored after exiting LP0.

Step 1

The Boot ROM enables USB related PLLs with software override.

The Boot ROM deasserts reset to XUSB PADCTL block.

The system software puts the system in LP0.

Step 2

The PAD driver switches USB related PLLs to under hardware control.

Refer to Step 2 of “Cold Boot”for the programming sequence.

Step 3

The PAD driver assigns the USB port to the controllers, then programs the port capabilities and pad parameters of ports assigned to XUSB according to the platform specific configuration.

Refer to Step 3 of “Cold Boot”for the programming sequence.

Step 4

The PAD driver programs the clocks and deasserts the resets to the controllers.

Refer to Step 4 of “Cold Boot”for the programming sequence.

- Wait 1 µs
- Program the following XUSB PADCTL register bits to ‘0’ to release the XUSB SS wake logic state latching
  - XUSB_PADCTL_ELPG_PROGRAM_0[SSP1_ELPG_VCORE_DOWN]
  - XUSB_PADCTL_ELPG_PROGRAM_0[SSP1_ELPG_VCORE_DOWN]
- Wait 100 µs.
- Program the following XUSB PADCTL register bits to ‘0’ to release the XUSB SS wake logic state latching
  - XUSB_PADCTL_ELPG_PROGRAM_0[SSP0_ELPG_CLAMP_EN_EARLY]
  - XUSB_PADCTL_ELPG_PROGRAM_0[SSP0_ELPG_CLAMP_EN_EARLY]
- Wait 100 µs
- Program the following XUSB PADCTL register bits to ‘0’ to release the XUSB SS wake logic state latching
  - XUSB_PADCTL_ELPG_PROGRAM_0[SSP0_ELPG_CLAMP_EN]
  - XUSB_PADCTL_ELPG_PROGRAM_0[SSP1_ELPG_CLAMP_EN]

Note: If XUSB was in ELPG before entering LP0 and the LP0 exit is not due to wake event from ports belong to XUSB, the PAD driver should keep XUSB in the ELPG state and only bring XUSB out of ELPG when wake events are detected for the ports owned by XUSB.

Step 5

The PAD driver enables the VBUS to the USB ports if VBUS was disabled during LP0.

Refer to Step 5 of “Cold Boot”for the programming sequence.

Step 6

The xHCI PEP driver performs XUSB specific IOPHY context restore for the following registers for the lanes assigned to XUSB based on the SS port mappings.

- Program the following XUSB PADCTL register bits with the context stored during the last time the port entered U0.
  - XUSB_PADCTL_IOPHY_MISC_PAD_P0_CTL_4_0[DFE_CNTL[28:24]]
  - XUSB_PADCTL_IOPHY_MISC_PAD_P0_CTL_4_0[DFE_CNTL[22:16]]
  - XUSB_PADCTL_IOPHY_MISC_PAD_P0_CTL_2_0[RX_EQ[13:8]]
  - XUSB_PADCTL_IOPHY_MISC_PAD_P0_CTL_2_0[RX_EQ[5:0]]
  - XUSB_PADCTL_IOPHY_MISC_PAD_P1_CTL_4_0[DFE_CNTL[28:24]]
  - XUSB_PADCTL_IOPHY_MISC_PAD_P1_CTL_4_0[DFE_CNTL[22:16]]
  - XUSB_PADCTL_IOPHY_MISC_PAD_P1_CTL_2_0[RX_EQ[13:8]]
  - XUSB_PADCTL_IOPHY_MISC_PAD_P1_CTL_2_0[RX_EQ[5:0]]
  - XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_4_0[DFE_CNTL[28:24]]
  - XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_4_0[DFE_CNTL[22:16]]
  - XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_2_0[RX_EQ[13:8]]
  - XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_2_0[RX_EQ[5:0]]

The xHCI PEP driver performs XUSB specific context restore operation

The xHCI PEP driver performs XUSB IPFS and XUSB register initialization as described in Step 6 of “Cold Boot”.

Step 7

The xHCI PEP Driver loads XUSB firmware.

Step 8

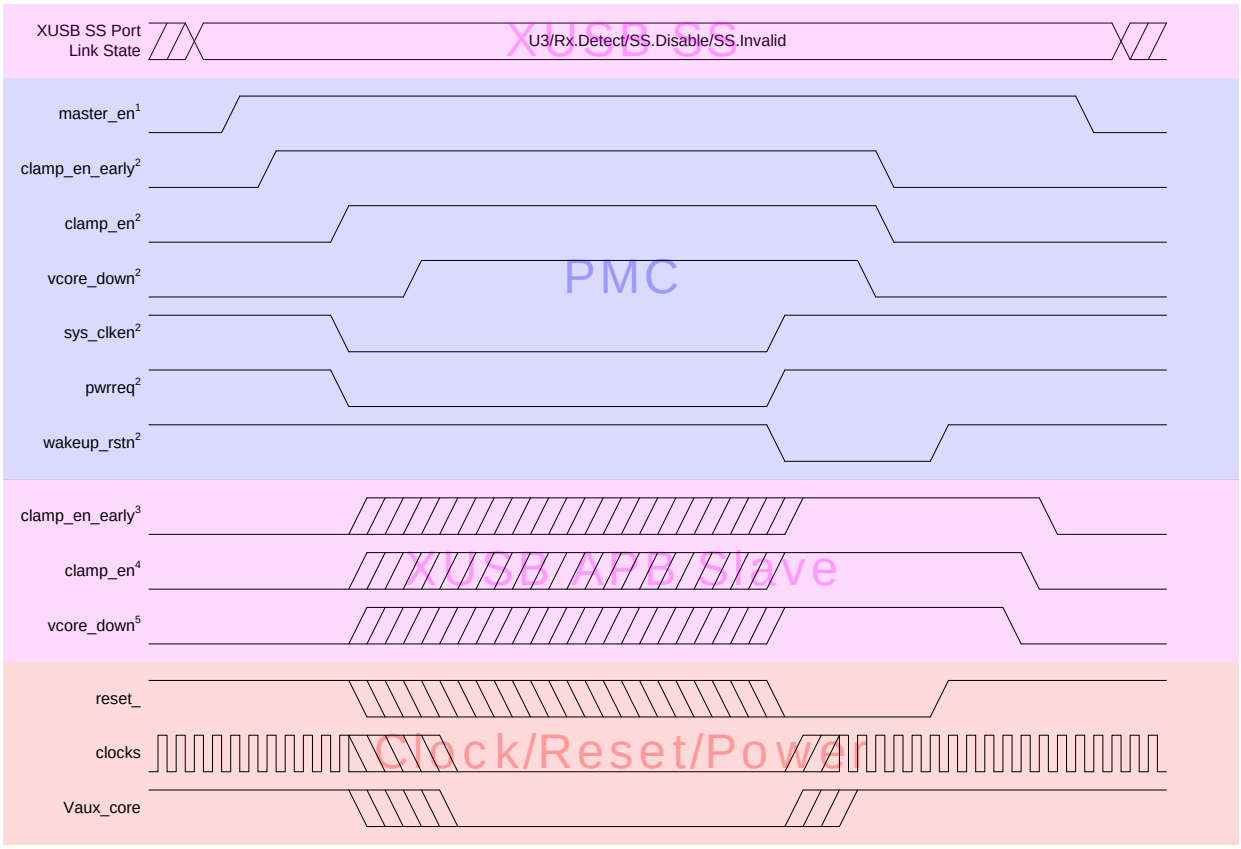
The xHCI driver performs context restore operation as described in section 4.23.2 of the xHCI specification,

Step 9

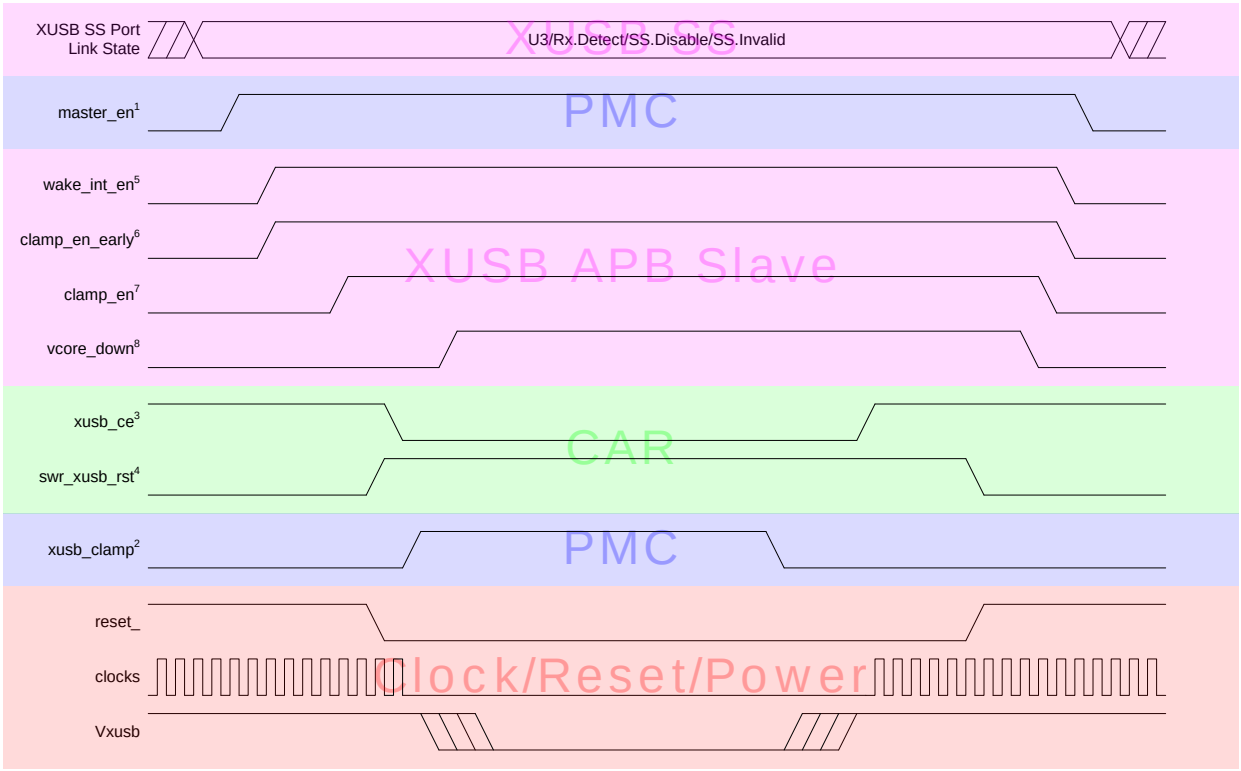
The System Power Management driver disables the PMC USB2.0 sleepwalk logic as described in the “PMC Programming” section.

19.7.7 XUSB Controller Power Gating

The following figure illustrates the signal sequences from the PMC that are used by XUSB SS Wake logic to enable its wake event detection.



The following figure illustrates the signal sequences from the XUSB PADCTL block that are required to follow the same sequence as from the PMC so XUSB SS Wake logic can enable its wake event detection.



The XUSB host controller has two partitions that can be selectively power gated with the following use cases, where unlisted combinations are not supported:

XUSBC (Host/USB2.0)	XUSBA (SuperSpeed)	Use Case Descriptions
Powered	Powered	Host controller in normal operations, with SuperSpeed links operational.
Powered	Power Gated	Host controller in normal operations, with SuperSpeed links in low power states. SuperSpeed link wake events report through XUSB PADCTL.
Power Gated	Power Gated	Host controllers power gated, with SuperSpeed links in low power states. All Host link wake events report through XUSB PADCTL.

19.7.7.1 All Partitions ELPG Entry

Step 1

The xHCI driver performs context save operation as described in section 4.23.2 of the xHCI specification.

The xHCI PEP driver performs XUSB_HOST specific context save operation.

The xHCI PEP driver performs XUSB IPFS specific context save operation.

- Read and store the value of the following registers
  - XUSB_HOST_MSI_BAR_SZ_0

- XUSB_HOST_MSI_AXI_BAR_ST_0
- XUSB_HOST_MSI_FPCI_BAR_ST_0
- XUSB_HOST_MSI_VEC0_0
- XUSB_HOST_MSI_EN_VEC0_0
- XUSB_HOST_FPCI_ERROR_MASKS_0
- XUSB_HOST_INTR_MASK_0
- XUSB_HOST_IPFS_INTR_ENABLE_0
- XUSB_HOST_UFPCI_CONFIG_0
- XUSB_HOST_CLKGATE_HYSTERESIS_0
- XUSB_HOST_XUSB_HOST_MCCIF_FIFOCTRL_0

Step 2

The System Power Management driver programs the PMC USB2.0 sleepwalk logic as described in the “PMC Programming” section.

The System Power Management driver should enable the wake events according to the port states and Wake-on-Connect, Wake-on-Disconnect, and Wake-on-Over-Current settings of the USB2 ports accordingly.

Step 3

The xHCI PEP driver enables the XUSB wakeup interrupts for the SuperSpeed and USB2.0 ports.

- Write ‘1’ to the following XUSB PADCTL register bits to clear the interrupt status.
  - XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT0_WAKEUP_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT1_WAKEUP_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT0_WAKEUP_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT1_WAKEUP_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT2_WAKEUP_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT0_WAKEUP_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT1_WAKEUP_EVENT]
- Set the following XUSB PADCTL register bits to ‘1’ to enable the interrupt.
  - XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT0_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT1_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT0_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT1_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT2_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT0_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT1_WAKE_INTERRUPT_ENABLE]
- Set the following XUSB PADCTL register bits to ‘1’ to disable the output driver of the HSIC pad.
  - XUSB_PADCTL_HSIC_PAD0_CTL_1_0[PD_TX]
  - XUSB_PADCTL_HSIC_PAD1_CTL_1_0[PD_TX]

Step 4

The xHCI PEP driver initiates the signal sequence to enable the XUSB SSwake detection logic for the SuperSpeed ports.

- Write '1' to the following XUSB PADCTL register bits to assert the clamp_en_early signal.
  - XUSB_PADCTL_ELPG_PROGRAM_0[SSP0_ELPG_CLAMP_EN_EARLY]
  - XUSB_PADCTL_ELPG_PROGRAM_0[SSP1_ELPG_CLAMP_EN_EARLY]
- Write '1' to the following XUSB PADCTL register bits to assert the clamp_en signal.
  - XUSB_PADCTL_ELPG_PROGRAM_0[SSP0_ELPG_CLAMP_EN]
  - XUSB_PADCTL_ELPG_PROGRAM_0[SSP1_ELPG_CLAMP_EN]
- Wait 250 µs

Note: These two writes must not be combined.

Step 5

System Power Management driver flushes MCCIF and partition clients.

- Set the following MC register bit to '1' to enable flush to XUSB
  - MC_CLIENT_HOTRESET_CTRL_0[XUSB_HOST_FLUSH_ENABLE] for host mode
- Read the following MC register bit to be '1' to ensure flush to XUSB is enabled
  - MC_CLIENT_HOTRESET_STATUS_0[XUSB_HOST_HOTRESET_STATUS] for host mode

Step 6

The System Power Management driver asserts reset to XUSB then disables its clocks.

- Set the following CAR register bits to '1' to assert reset to XUSB
  - CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_HOST_RST] for host mode
  - CLK_RST_CONTROLLER_RST_DEVICES_W_0[SWR_XUSB_SS_RST] for SuperSpeed ports
- Set the following CAR register bits to '1' to disable the clocks to individual XUSB partitions.
  - CLK_RST_CONTROLLER_CLK_ENB_U_CLR_0[CLR_CLK_ENB_XUSB_HOST] for host mode
  - CLK_RST_CONTROLLER_CLK_ENB_W_CLR_0[CLR_CLK_ENB_XUSB_SS] for SS ports

Step 7

The System Power Management driver disables the XUSB power rails.

- Program the following PMC register bits in a single write to disable the power rail to XUSB host:
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to 'enable'
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to 'XUSBC'
- Read the following PMC register bit to confirm the power gating status of XUSB host
  - APBDEV_PMC_PWRGATE_STATUS_0[XUSBC] equals 'OFF'
- Program the following PMC register bits in a single write to disable the power rail to XUSB SuperSpeed:
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to 'enable'
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to 'XUSBA'
- Read the following PMC register bit to confirm the power gating status of XUSB SuperSpeed:

- APBDEV_PMC_PWRGATE_STATUS_0[XUSBA] equals ‘OFF’

Note: Only one partition can be power gated at a time

Step 8

The xHCI PEP driver initiates the signal sequence to enable the XUSB SS wake detection logic for the SuperSpeed ports assigned to host.

- Write ‘1’ to the following XUSB PADCTL register bits to assert the vcore_off signal:
  - XUSB_PADCTL_ELPG_PROGRAM_0[SSP0_ELPG_VCORE_DOWN]
  - XUSB_PADCTL_ELPG_PROGRAM_0[SSP1_ELPG_VCORE_DOWN]

19.7.7.2 Host Mode Partition ELPG Entry

Step 1

The xHCI driver performs context save operation as described in section 4.23.2 of the xHCI specification.

The xHCI PEP driver performs XUSB_HOST specific context save operation.

The xHCI PEP driver performs XUSB IPFS specific context save operation.

Step 2

The System Power Management driver programs the PMC USB2.0 sleepwalk logic as described in the “PMC Programming” section.

The System Power Management driver should enable the wake events according to the port states and Wake-on-Connect, Wake-on-Disconnect, and Wake-on-Over-Current settings of the USB2 ports accordingly.

Step 3

The xHCI PEP driver enables the XUSB wakeup interrupts for the SuperSpeed and USB2.0 ports.

Step 4

The System Power Management driver flushes MCCIF and partition clients.

- Set the following MC register bit to ‘1’ to enable flush to XUSB:
  - MC_CLIENT_HOTRESET_CTRL_0[XUSB_HOST_FLUSH_ENABLE] for host mode
- Read the following MC register bit to be ‘1’ to ensure flush to XUSB is enabled:
  - MC_CLIENT_HOTRESET_STATUS_0[XUSB_HOST_HOTRESET_STATUS] for host mode

Step 5

The System Power Management driver asserts reset to XUSB then disables its clocks.

- Set the following CAR register bit to ‘1’ to assert reset to XUSB
  - CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_HOST_RST] for host mode
- Set the following CAR register bits to ‘1’ to disable the clocks to individual XUSB partitions.
  - CLK_RST_CONTROLLER_CLK_ENB_U_CLR_0[CLR_CLK_ENB_XUSB_HOST] for host mode

Step 6

The System Power Management driver disables the XUSB power rails.

- Program the following PMC register bits in a single write to disable the power rail to XUSB host:

- APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
- APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘XUSBC’

- Read the following PMC register bit to confirm the power gating status of XUSB host:
  - APBDEV_PMC_PWRGATE_STATUS_0[XUSBC] equals ‘OFF’

Step 7

The xHCI PEP driver moves the port to be under USB2 controllers as a workaround to use USB2 controller outputs to put the USB2.0 PADs in low power mode.

- Program ‘SNPS’ to the following XUSB PADCTL register bits for ports assigned to XUSB:
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT2]
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT1]
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT0]

19.7.7.3 SuperSpeed Partition ELPG Entry

Step 1

The xHCI PEP driver communicates SuperSpeed partition ELPG entry with the System Power Management driver.

Step 2

The xHCI PEP driver enables the XUSB wakeup interrupts for the SuperSpeed ports.

Step 3

The xHCI PEP driver initiates the signal sequence to enable the XUSB SSwake detection logic for the SuperSpeed ports as described in Step 4 of the “All Partitions ELPG Entry” section.

Step 4

The System Power Management driver asserts reset to XUSB SuperSpeed partition then disables its clocks.

- Set the following CAR register bit to ‘1’ to assert reset to XUSB:
  - CLK_RST_CONTROLLER_RST_DEVICES_W_0[SWR_XUSB_SS_RST] for SuperSpeed ports
- Set the following CAR register bit to ‘1’ to disable the clocks to individual XUSB partitions:
  - CLK_RST_CONTROLLER_CLK_ENB_W_CLR_0[CLR_CLK_ENB_XUSB_SS] for SS ports

Step 5

The System Power Management driver disables the XUSB SuperSpeed partition power rails.

- Program the following PMC register bits in a single write to disable the power rail to XUSB SuperSpeed:
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘XUSBA’
- Read the following PMC register bit to confirm the power gating status of the partitions:
  - APBDEV_PMC_PWRGATE_STATUS_0[XUSBA] equals ‘OFF’

Note: Only one partition can be power gated at a time.

Step 6

The xHCI PEP driver initiates the signal sequence to enable the XUSB SS wake detection logic for the SuperSpeed ports as described in Step 7 of the “All Partitions ELPG Entry” section.

19.7.7.4 All Partitions ELPG Exit

In the case the wake event is not generated by SuperSpeed ports, the SuperSpeed related programming should be skipped, where the SuperSpeed partition should exit ELPG due to wake events detected by SuperSpeed ports.

Step 1

The xHCI PEP driver moves the USB2.0 port back to XUSB.

- Program the following XUSB PADCTL registers to assign the USB2.0 ports to XUSB, according to the platform specific configuration:
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT2]
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT1]
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT0]

Step 2

The System Power Management driver enables the XUSB power rails.

- Program the following PMC register bits in a single write to enable the power rail to XUSB host:
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘XUSBC’
- Read the following PMC register bit to confirm the power gating status of XUSB host:
  - APBDEV_PMC_PWRGATE_STATUS_0[XUSBC] equals ‘ON’
- Program the following PMC register bits in a single write to disable the power rail to XUSB SuperSpeed:
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘XUSBA’
- Read the following PMC register bit to confirm the power gating status of XUSB SuperSpeed:
  - APBDEV_PMC_PWRGATE_STATUS_0[XUSBA] equals ‘ON’

Note: Only one partition can be un-power gated at a time.

Step 3

The System Power Management driver enables XUSB clocks.

- Set the following CAR register bits to ‘1’ to enable the clocks to individual XUSB partitions.
  - CLK_RST_CONTROLLER_CLK_ENB_U_SET_0[SET_CLK_ENB_XUSB_HOST] for host mode
  - CLK_RST_CONTROLLER_CLK_ENB_W_SET_0[SET_CLK_ENB_XUSB_SS] for SS ports

Step 4

The System Power Management driver removes power clamps to XUSB partitions.

- Set the following PMC register bits to ‘1’ to remove the power clamps to individual XUSB partitions.
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [XUSBC] for host mode

- APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [XUSBA] for SS ports

- Read the following PMC register bits to confirm the power clamps to individual XUSB partitions are removed.
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [XUSBC] equals '0'
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [XUSBA] equals '0'

Step 5

The System Power Management driver deasserts reset to XUSB.

- Set the following CAR register bits to '0' to deassert reset to XUSB:
  - CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_HOST_RST] for host mode
  - CLK_RST_CONTROLLER_RST_DEVICES_W_0[SWR_XUSB_SS_RST] for SuperSpeed ports
- Wait 1 µs

Step 6

The System Power Management driver disables flushes of MCCIF and partition clients.

- Set the following MC register bits to '0' to enable flush to XUSB:
  - MC_CLIENT_HOTRESET_CTRL_0[XUSB_HOST_FLUSH_ENABLE] for host mode

Step 7

The xHCI PEP driver disables the XUSB wakeup interrupts.

- Set the following XUSB PADCTL register bits to '0' to disable the interrupts.
  - XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT0_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT1_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT0_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT1_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT2_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT0_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT1_WAKE_INTERRUPT_ENABLE]

Step 8

The xHCI PEP driver initiates the signal sequence to disable the XUSB SS wake detection logic.

- Write '0' to the following XUSB PADCTL register bits to deassert the vcore_off signal.
  - XUSB_PADCTL_ELPG_PROGRAM_0[SSP0_ELPG_VCORE_DOWN]
  - XUSB_PADCTL_ELPG_PROGRAM_0[SSP1_ELPG_VCORE_DOWN]
- Wait 100 µs.
- Write '0' to the following XUSB PADCTL register bits to deassert the clamp_en_early signals.
  - XUSB_PADCTL_ELPG_PROGRAM_0[SSP0_ELPG_CLAMP_EN_EARLY]
  - XUSB_PADCTL_ELPG_PROGRAM_0[SSP1_ELPG_CLAMP_EN_EARLY]
- Wait 100 µs.
- Write '0' to the following XUSB PADCTL register bits to deassert the clamp_en signals.
  - XUSB_PADCTL_ELPG_PROGRAM_0[SSP0_ELPG_CLAMP_EN]

- XUSB_PADCTL_ELPG_PROGRAM_0[SSP1_ELPG_CLAMP_EN]

Note: The write to clear vcore_off cannot be combined with the write to clear clamp_en and clamp_en_early.

Step 9

The xHCI PEP driver performs XUSB register initialization.

The xHCI PEP driver performs XUSB context restore operation.

Step 10

The xHCI PEP driver loads XUSB firmware.

The xHCI PEP driver notifies XUSB firmware whether context of SuperSpeed Partition should be restored.

Step 11

The xHCI driver performs context restore operation as described in section 4.23.2 of the xHCI specification.

Step 12

The System Power Management driver programs the PMC USB2.0 sleepwalk logic to disable the sleepwalk logic as described in the “PMC Programming” section.

The xHCI PEP driver clears the XUSB wakeup events.

- Write ‘1’ to the following XUSB PADCTL register bits to clear the events.
  - XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT0_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT1_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT0_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT1_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT2_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT0_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT1_WAKE_EVENT]

19.7.7.5 Host Mode ELPG Exit

Step 1

The xHCI PEP driver driver moves the USB2.0 port back to XUSB.

- Program the following XUSB PADCTL registers to assign the USB2.0 ports to XUSB, according to the platform specific configuration:
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT2]
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT1]
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT0]

Step 2

The System Power Management driver enables the XUSB power rails.

- Program the following PMC register bits in a single write to enable the power rail to XUSB host:
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘XUSBC’

- Read the following PMC register bit to confirm the power gating status of XUSB host:
  - APBDEV_PMC_PWRGATE_STATUS_0[XUSBC] equals ‘ON’

Step 3

The System Power Management driver enables XUSB clocks.

- Set the following CAR register bit to ‘1’ to enable the clocks to individual XUSB partitions:
  - CLK_RST_CONTROLLER_CLK_ENB_U_SET_0[SET_CLK_ENB_XUSB_HOST] for host mode

Step 4

The System Power Management driver removes power clamps to XUSB partitions.

- Set the following PMC register bit to ‘1’ to remove the power clamps to individual XUSB partitions:
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [XUSBC] for host mode
- Read the following PMC register bits to confirm the power clamps to individual XUSB partitions are removed.
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [XUSBC] equals ‘0’

Step 5

The System Power Management driver deasserts reset to XUSB.

- Set the following CAR register bit to ‘0’ to deassert reset to XUSB
  - CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_HOST_RST] for host mode
- Wait 1 µs

Step 6

The System Power Management driver disables flushes of MCCIF and partition clients.

- Set the following MC register bits to ‘0’ to enable flush to XUSB
  - MC_CLIENT_HOTRESET_CTRL_0[XUSB_HOST_FLUSH_ENABLE] for host mode

Step 7

The xHCI PEP driver disables the USB2.0wakeup interrupts for ports assigned to XUSB.

- Set the following XUSB PADCTL register bits to ‘0’ to disable the interrupts.
  - XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT0_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT1_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT0_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT1_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT2_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT0_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT1_WAKE_INTERRUPT_ENABLE]

Step 8

The xHCI PEP driver performs XUSB register initialization.

The xHCI PEP driver performs XUSB specific context restore operation.

Step 9

The xHCI PEP Driver loads XUSB firmware.

The xHCI PEP Driver notifies XUSB firmware whether context of SuperSpeed Partition should be restored.

Step 10

The xHCI driver performs context restore operation as described in section 4.23.2 of the xHCI specification.

Step 11

The System Power Management driver programs the PMC USB2.0 sleepwalk logic to disable the sleepwalk logic as described in the “PMC Programming” section.

The xHCI PEP driver clears the XUSB wakeup events.

- Write ‘1’ to the following XUSB PADCTL register bits to clear the events.
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT0_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT1_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT2_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT0_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT1_WAKE_EVENT]

19.7.7.6 SuperSpeed Partition ELPG Exit

Step 1

The xHCI PEP driver communicates SuperSpeed partition ELPG exit with System Power Management driver.

Step 2

The System Power Management driver enables the XUSB power rails.

- Program the following PMC register bits in a single write to disable the power rail to XUSB SuperSpeed
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘XUSBA’
- Read the following PMC register bits to confirm the power gating status of XUSB SuperSpeed
  - APBDEV_PMC_PWRGATE_STATUS_0[XUSBA] equals ‘ON’

Note: Only one partition can be un-power gated at a time

Step 3

The System Power Management driver enables XUSB clocks.

- Set the following CAR register bits to ‘1’ to enable the clocks to individual XUSB partitions.
  - CLK_RST_CONTROLLER_CLK_ENB_W_SET_0[SET_CLK_ENB_XUSB_SS] for SS ports

Step 4

The xHCI PEP driver initiates the signal sequence to disable the XUSB SS wake detection logic for the SuperSpeed ports as described in Step 3 of the “All Partitions ELPG Exit” section.

Step 5

The xHCI PEP driver disable the XUSB SS wakeup interrupts.

- Set the following XUSB PADCTL register bits to ‘0’ to disable the interrupts.
  - XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT0_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT1_WAKE_INTERRUPT_ENABLE]

Step 6

The System Power Management driver removes power clamps to XUSB partitions.

- Set the following PMC register bit to ‘1’ to remove the power clamp to XUSB SS:
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [XUSBA]
- Read the following PMC register bits to confirm the power clamp to XUSB SS is removed.
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [XUSBA] equals ‘0’

Step 7

The System Power Management driver deasserts reset to XUSB.

- Set the following CAR register bit to ‘0’ to deassert reset to XUSB:
  - CLK_RST_CONTROLLER_RST_DEVICES_W_0[SWR_XUSB_SS_RST] for SuperSpeed ports

Step 8

The xHCI PEP driver clears the XUSB wakeup events.

- Write ‘1’ to the following XUSB PADCTL register bits to clear the events.
  - XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT0_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT1_WAKE_EVENT]

19.7.8 PMC Programming

Sleepwalk logic in the PMC is used for wake event detection of USB2.0 ports, including wake on connect, wake on disconnect, and remote wakeup.

19.7.8.1 Initialize PMC Sleepwalk Logic

Step 1

The xHCI PEP driver initializes the sleepwalk logic:

- Set the following registers to ‘0’ to ensure sleepwalk logic is disabled:
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_MASTER_ENABLE_P3]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UHSIC_MASTER_ENABLE_P0]
- Set the following registers to ‘1’ to ensure sleepwalk logic is in low power mode:
  - APBDEV_PMC_UTMIP_MASTER_CONFIG_0[UTMIP_PWR_P0]
  - APBDEV_PMC_UTMIP_MASTER_CONFIG_0[UTMIP_PWR_P1]
  - APBDEV_PMC_UTMIP_MASTER_CONFIG_0[UTMIP_PWR_P2]
  - APBDEV_PMC_UTMIP_MASTER_CONFIG_0[UTMIP_PWR_P3]

- APBDEV_PMC_UTMIP_MASTER_CONFIG_0[UHSIC_PWR_P0]

- Program the following registers to set debounce time:
  - APBDEV_PMC_USB_DEBOUNCE_DEL_0[UTMIP_LINE_DEB_CNT]
  - APBDEV_PMC_USB_DEBOUNCE_DEL_0[UHSIC_LINE_DEB_CNT]
- Set the following registers to ‘0’ to ensure fake events of sleepwalk logic are disabled:
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBOP_EN_P0]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBON_EN_P0]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBOP_VAL_P0]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBON_VAL_P0]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBOP_EN_P1]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBON_EN_P1]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBOP_VAL_P1]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBON_VAL_P1]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBOP_EN_P2]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBON_EN_P2]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBOP_VAL_P2]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBON_VAL_P2]
  - APBDEV_PMC_UTMIP_UHSIC2_FAKE_0[UTMIP_FAKE_USBOP_EN_P3]
  - APBDEV_PMC_UTMIP_UHSIC2_FAKE_0[UTMIP_FAKE_USBON_EN_P3]
  - APBDEV_PMC_UTMIP_UHSIC2_FAKE_0[UTMIP_FAKE_USBOP_VAL_P3]
  - APBDEV_PMC_UTMIP_UHSIC2_FAKE_0[UTMIP_FAKE_USBON_VAL_P3]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UHSIC_FAKE_DATA_EN_P0]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UHSIC_FAKE_STROBE_EN_P0]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UHSIC_FAKE_DATA_VAL_P0]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UHSIC_FAKE_STROBE_VAL_P0]
- Set the following registers to ‘0’ to ensure wake events of sleepwalk logic are not latched:
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P0]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P1]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P2]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P3]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UHSIC_LINE_WAKEUP_EN_P0]
- Program the following registers to ‘NONE’ to disable wake event triggers of sleepwalk logic:
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_WAKE_VAL_P3]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UHSIC_WAKE_VAL_P0]

- Set the following registers to ‘0’ to power down the line state detectors of the pad:
  - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P0]
  - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P0]
  - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P1]
  - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P1]
  - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P2]
  - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P2]
  - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P3]
  - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P3]
  - APBDEV_PMC_USB_AO_0[DATA0_VAL_PD_P0]
  - APBDEV_PMC_USB_AO_0[DATA1_VAL_PD_P0]
  - APBDEV_PMC_USB_AO_0[STROBE_VAL_PD_P0]

19.7.8.2 Enable PMC Sleepwalk Logic

Step 1

The xHCI PEP driver re-initializes the sleepwalk logic as described in the “Initialize PMC Sleepwalk Logic” section.

Step 2

The xHCI PEP driver sets up the sleepwalk logic.

- Program the following registers to match the speed of the port:
  - APBDEV_PMC_UTMIP_UHSIC_SAVED_STATE_0[UTMIP_SPEED_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SAVED_STATE_0[UTMIP_SPEED_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SAVED_STATE_0[UTMIP_SPEED_P2]
  - APBDEV_PMC_UTMIP_UHSIC2_SAVED_STATE_0[UTMIP_SPEED_P3]
  - APBDEV_PMC_UTMIP_UHSIC_SAVED_STATE_0[UHSIC_SPEED_P0]
- Set the following registers to ‘1’ to enable the trigger of the sleepwalk logic:
  - APBDEV_PMC_UTMIP_UHSIC_SLEEPWALK_CFG_0[UTMIP_WAKE_WALK_EN_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEPWALK_CFG_0[UTMIP_LINEVAL_WALK_EN_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEPWALK_CFG_0[UTMIP_WAKE_WALK_EN_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEPWALK_CFG_0[UTMIP_LINEVAL_WALK_EN_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEPWALK_CFG_0[UTMIP_WAKE_WALK_EN_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEPWALK_CFG_0[UTMIP_LINEVAL_WALK_EN_P2]
  - APBDEV_PMC_UTMIP_UHSIC2_SLEEPWALK_CFG_0[UTMIP_WAKE_WALK_EN_P3]
  - APBDEV_PMC_UTMIP_UHSIC2_SLEEPWALK_CFG_0[UTMIP_LINEVAL_WALK_EN_P3]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEPWALK_CFG_0[UHSIC_WAKE_WALK_EN_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEPWALK_CFG_0[UHSIC_LINEVAL_WALK_EN_P0]
- Write ‘1’ to the following registers to reset the walk pointer and clear the alarm of the sleepwalk logic, as well as capture the configuration of the USB2.0 pad:
  - APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WALK_PTR_P0]

- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WAKE_ALARM_P0]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CAP_CFG_P0]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WALK_PTR_P1]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WAKE_ALARM_P1]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CAP_CFG_P1]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WALK_PTR_P2]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WAKE_ALARM_P2]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CAP_CFG_P2]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WALK_PTR_P3]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WAKE_ALARM_P3]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CAP_CFG_P3]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UHSIC_CLR_WALK_PTR_P0]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UHSIC_CLR_WAKE_ALARM_P0]

- Program the following registers with electrical parameters read from XUSB PADCTL:
  - APBDEV_PMC_UTMIP_TERM_PAD_CFG_0[TCTRL_VAL] to XUSB_PADCTL_USB2_BIAS_PAD_CTL_1_0[TCTRL]
  - APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[PCTRL_VAL] to XUSB_PADCTL_USB2_BIAS_PAD_CTL_1_0[PCTRL]
- Program the following registers to set up the pull-ups and pull-downs of the signals during the four stages of sleepwalk:
  - APBDEV_PMC_UTMIP_SLEEPWALK_P0_0[USBOP_RPD_{A/B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P0_0[USBON_RPD_{A/B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P0_0[USBOP_RPU_{A/B/C/D}] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P0_0[USBON_RPU_{A/B/C/D}] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P1_0[USBOP_RPD_{A/B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P1_0[USBON_RPD_{A/B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P1_0[USBOP_RPU_{A/B/C/D}] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P1_0[USBON_RPU_{A/B/C/D}] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P2_0[USBOP_RPD_{A/B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P2_0[USBON_RPD_{A/B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P2_0[USBOP_RPU_{A/B/C/D}] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P2_0[USBON_RPU_{A/B/C/D}] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P3_0[USBOP_RPD_{A/B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P3_0[USBON_RPD_{A/B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P3_0[USBOP_RPU_{A/B/C/D}] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P3_0[USBON_RPU_{A/B/C/D}] to ‘0’
  - APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[STROBE_RPD_A] to ‘0’
  - APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[DATA0_RPD_A] to ‘1’
  - APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[DATA1_RPD_A] to ‘1’

- APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[STROBE_RPU_A] to ‘1’
- APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[DATA0_RPU_A] to ‘0’
- APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[DATA1_RPU_A] to ‘0’
- APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[STROBE_RPD_{B/C/D}] to ‘1’
- APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[DATA0_RPD_{B/C/D}] to ‘0’
- APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[DATA1_RPD_{B/C/D}] to ‘1’
- APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[STROBE_RPU_{B/C/D}] to ‘0’
- APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[DATA0_RPU_{B/C/D}] to ‘1’
- APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[DATA1_RPU_{B/C/D}] to ‘0’

- Program the following registers to set up the driving values of the signals during the four stages of sleepwalk:
  - APBDEV_PMC_UTMIP_SLEEPWALK_P0_0[AP_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P0_0[AN_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P0_0[HIGNZ_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P0_0[AP_{B/C/D}] to ‘0’ for HS/FS and ‘1’ for LS
  - APBDEV_PMC_UTMIP_SLEEPWALK_P0_0[AN_{B/C/D}] to ‘1’ for HS/FS and ‘0’ for LS
  - APBDEV_PMC_UTMIP_SLEEPWALK_P0_0[HIGHZ_{B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P1_0[AP_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P1_0[AN_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P1_0[HIGNZ_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P1_0[AP_{B/C/D}] to ‘0’ for HS/FS and ‘1’ for LS
  - APBDEV_PMC_UTMIP_SLEEPWALK_P1_0[AN_{B/C/D}] to ‘1’ for HS/FS and ‘0’ for LS
  - APBDEV_PMC_UTMIP_SLEEPWALK_P1_0[HIGHZ_{B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P2_0[AP_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P2_0[AN_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P2_0[HIGNZ_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P2_0[AP_{B/C/D}] to ‘0’ for HS/FS and ‘1’ for LS
  - APBDEV_PMC_UTMIP_SLEEPWALK_P2_0[AN_{B/C/D}] to ‘1’ for HS/FS and ‘0’ for LS
  - APBDEV_PMC_UTMIP_SLEEPWALK_P2_0[HIGHZ_{B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P3_0[AP_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P3_0[AN_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P3_0[HIGNZ_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P3_0[AP_{B/C/D}] to ‘0’ for HS/FS and ‘1’ for LS
  - APBDEV_PMC_UTMIP_SLEEPWALK_P3_0[AN_{B/C/D}] to ‘1’ for HS/FS and ‘0’ for LS
  - APBDEV_PMC_UTMIP_SLEEPWALK_P3_0[HIGHZ_{B/C/D}] to ‘1’

Note: For HSIC and HSIS+, DATA1 is not used for signaling wake.

Step 3

The xHCI driver puts the ports in the U3 suspend state.

Step 4

The xHCI PEP driver enables the wake event detections.

- Set the following registers to '1' to power up the line state detectors of the pad
  - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P0]
  - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P0]
  - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P1]
  - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P1]
  - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P2]
  - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P2]
  - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P3]
  - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P3]
  - APBDEV_PMC_USB_AO_0[DATA0_VAL_PD_P0]
  - APBDEV_PMC_USB_AO_0[DATA1_VAL_PD_P0]
  - APBDEV_PMC_USB_AO_0[STROBE_VAL_PD_P0]
- Wait 1 µs
- Set the following registers to '1' to switch the electric control of the USB2.0 pad to PMC
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_FSLS_USE_PMC_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_PCTRL_USE_PMC_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_TCTRL_USE_PMC_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_FSLS_USE_PMC_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_PCTRL_USE_PMC_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_TCTRL_USE_PMC_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_FSLS_USE_PMC_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_PCTRL_USE_PMC_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_TCTRL_USE_PMC_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_FSLS_USE_PMC_P3]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_PCTRL_USE_PMC_P3]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_TCTRL_USE_PMC_P3]
- Program the following registers to set the wake signaling trigger events
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P0] to 'ANY'
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P1] to 'ANY'
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P2] to 'ANY'
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_WAKE_VAL_P3] to 'ANY'
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UHSIC_WAKE_VAL_P0] to 'SD10'
- Set the following registers to '1' to enable the wake detection
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P0]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P0]

- APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P1]
- APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P1]
- APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P2]
- APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P2]
- APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_MASTER_ENABLE_P3]
- APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P3]
- APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UHSIC_MASTER_ENABLE_P0]
- APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UHSIC_LINE_WAKEUP_EN_P0]

Note: For USB2 ports, set wake value to ANY means any line state change would trigger wake.

Note: For HSIC and HSIS+, DATA1 is not used for wake detection.

19.7.8.3 Disable PMC Sleepwalk Logic

Step 1

The xHCI PEP driver disables the wake event detections.

- Set the following registers to '0' to disable the wake detection:
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P0]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P1]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P2]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_MASTER_ENABLE_P3]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P3]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UHSIC_MASTER_ENABLE_P0]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UHSIC_LINE_WAKEUP_EN_P0]
- Set the following registers to '0' to switch the electric control of the USB2.0 pad to XUSB or USB2:
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_FSLS_USE_PMC_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_PCTRL_USE_PMC_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_TCTRL_USE_PMC_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_FSLS_USE_PMC_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_PCTRL_USE_PMC_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_TCTRL_USE_PMC_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_FSLS_USE_PMC_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_PCTRL_USE_PMC_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_TCTRL_USE_PMC_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_FSLS_USE_PMC_P3]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_PCTRL_USE_PMC_P3]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_TCTRL_USE_PMC_P3]

- Program the following registers to ‘NONE’ to disable wake event triggers of sleepwalk logic:
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_WAKE_VAL_P3]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UHSIC_WAKE_VAL_P0]
- Set the following registers to ‘0’ to power down the line state detectors of the pad:
  - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P0]
  - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P0]
  - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P1]
  - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P1]
  - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P2]
  - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P2]
  - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P3]
  - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P3]
  - APBDEV_PMC_USB_AO_0[DATA0_VAL_PD_P0]
  - APBDEV_PMC_USB_AO_0[DATA1_VAL_PD_P0]
  - APBDEV_PMC_USB_AO_0[STROBE_VAL_PD_P0]
- Write ‘1’ to the following registers to clear alarm of the sleepwalk logic:
  - APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WAKE_ALARM_P0]
  - APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WAKE_ALARM_P1]
  - APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WAKE_ALARM_P2]
  - APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WAKE_ALARM_P3]
  - APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UHSIC_CLR_WAKE_ALARM_P0]

19.8 Recommended PHY Settings

Contact your local NVIDIA representative for PHY settings specific to your design.

19.9 BIAS PAD Configuration

The UTMIP BIAS pad is shared across USB_OTG and USB3 controllers and the below common bias pad settings need to be configured from the USB1 controller only.

USB1_UTMIP_BIAS_CFG0_0:

- UTMIP_BIASPD
- UTMIP_HSCHIRP_LEVEL
- UTMIP_HSSQUELCH_LEVEL
- UTMIP_HSDISCON_LEVEL_MSB
- UTMIP_HSDISCON_LEVEL

- UTMIP_ACTIVE_TERM_OFFSET
- UTMIP_ACTIVE_PULLUP_OFFSET
- UTMIP_VBUS_LEVEL_LEVEL
- UTMIP_SESS_LEVEL_LEVEL

USB1_UTMIP_BIAS_CFG1_0:

- UTMIP_FORCE_PDTRK_POWERUP
- UTMIP_FORCE_PDTRK_POWERDOWN

Following is the sequence to configure above parameters of the BIAS pad:

1. Enable the clock to the USB1 controller
2. Set any of above parameters as required.
3. Disable the clock to the USB1 controller if it is not used.

19.10 BOOT ROM Initialization Sequence for USB Recovery

1. Program PLL_U.
   - Set the PLLU_BASE register fields, PLLU_DIVM, PLLU_DIVN, PLLU_VCO_FREQ, PLLU_BYPASS and PLLU_ENABLE fields as described in this document.
2. Configure USB_OTG
3. Bring up USB_OTG clocks by writing 1 to CLK_ENB_USBD in the CLK_OUT_ENB_L register.
4. Assert and deassert the master USBD reset in the CAR block (SWR_USBD_RST in the RST_DEVICES_L register) to bring USB_OTG out of reset.
5. Stop the crystal clock by setting UTMIP_PHY_XTAL_CLOCKEN in the UTMIP_MISC_CFG1 register to 0. This only stops the crystal clocks in the UTMIP units.
   - The default value of USB1_UTMIP_PHY_XTAL_CLOCKEN (= 1) now changes to 0.
   - To use the A Session Valid for cable detection logic, set the USB1_VBUS_SENSE_CTL field in the USB1_LEGACY_CTRL register to A_SESS_VLD (2'b11).
6. Program the automatic PLL start times.

   Set 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 this document.
7. Program the tracking duration.

   Set USB1_IF_UTMIP_BIAS_CFG1.UTMIP_BIAS_PDTRK_COUNT as per the values given in this document.
8. Program the debouncer length times.

   Set UTMIP_DEBOUNCE_CFG0.UTMIP_BIAS_DEBOUNCE_A field.
9. Program various static parameters of the USB_OTG 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 the third bit of UTMIP_SPARE_CFG0 to 1. i.e., UTMIP_SPARE_CFG0[3] to 1.
- Set UTMIP_HSRX_CFG0. UTMIP_IDLE_WAIT as per the values given in this document.
- Set UTMIP_HSRX_CFG0.UTMIP_ELASTIC_LIMIT to 16.
- Set UTMIP_HSRX_CFG1.UTMIP_HS_SYNC_START_DLY to 9

10. Restart the crystal clock by setting UTMIP_PHY_XTAL_CLOCKEN in the UTMIP_MISC_CFG1 register to 1 for USB_OTG.
11. Wait for cable connect on the USB_OTG UTMIP1 port. When cable is connected on USB_OTG UTMIP1 port, continue to the next step.
12. Bring UTMIP1 out of reset by writing 0 to the UTMIP_RESET bit of the USB1_IF_SUSP_CTRL register.
13. Wait until USB1_IF_USB_SUSP_CTRL.PHY_CLK_VALID is set to 1.
14. Then perform USB controller initialization.
    a. Reset the bus.
15. Wait until the bus comes out of reset.
16. Set the controller in device mode.
17. Perform USB operations.

19.11 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 the USB controller should be programmed to 8 in the USB2D_BURSTSIZE register. Both TXPBURST and RXPBURST fields should be programmed to the same value of 8.
- The ENB_FAST_REARBITRATE field for AHB_MEM gizmo should be set to 1 in the AHB_GIZMO_AHB_MEM register.
- The IMMEDIATE field for USB gizmos should be set to 1 in the AHB_GIZMO_USB, AHB_GIZMO_USB2, or AHB_GIZMO_USB3 register, 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 AHB_PRIORITY_WEIGHT field in the same register. USB master numbers are 6 for USB_OTG, 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.
- The prefetch engine needs to be set up 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 AHB_AHB_MEM_PREFETCH_CFG[NO] register, 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 AHB_MST_ID field for the AHB_AHB_MEM_PREFETCH_CFG[NO] register. The ADDR_BNDRY field should be set to log2 (buffer size) according to the buffer size required for the corresponding USB master. The SPEC_THROTTLE field should be set to 0, and the INACTIVITY_TIMEOUT field should be set to 0x800.
- When a particular USB controller is in Host mode, the TXFIFOTHRES field in the USB2D_TXFILLTUNING register (offset 0x154) should be set to 0x10.

All this programming needs to be done before the RS bit in USB2D_USBCMD is set to RUN (1) for the corresponding USB controller.

19.12 USB Controller Handling of USB Resume Sequence

Following are the steps for how the USB controller handles a USB resume while the port is under PMC control:

1. The PMC maintains suspend mode on the bus.
2. When auto resume happens, the system starts restoring the USB controller.
3. Set the RUN bit from the USB controller to start SOFs.
4. The USB controller is brought back to suspend state.
5. Switch the USB bus from the PMC to the USB controller.
6. Wait until a resume complete notification from the USB controller. This is handled by the EHCI driver.
7. Further USB communication can start from here.

19.13 USB Registers

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

19.13.1 USB 1 Controller Registers

19.13.1.1 USB2_CONTROLLER_USB2D_ID_0

USB2D Identification Register

Offset: 0x0 | Read/Write: RO |Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	CIVERSION: Identifies the CI version
28:25	X	VERSION: Identifies the version of the core
24:21	X	REVISION: Revision number of the USB controller. This is set to 0x0.
20:16	X	TAG: Identifies the tag of the core
15:8	X	NID: One’s complement version of ID. This field is set to 0xF9.
7:0	X	ID: Configuration number. This field is set to 0x06

19.13.1.2 USB2_CONTROLLER_USB2D_HW_HOST_0

USB2D Hardware Host Register

Offset: 0x8 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
3:1	X	NPORT: VUSB_HS_NUM_PORT-1: This host controller has only 1 port. So this field will always be 0.
0	X	HC: VUSB_HS_HOST: Indicates support for host mode. Set to 1.

19.13.1.3 USB2_CONTROLLER_USB2D_HW_DEVICE_0

USB2D Hardware Device Register

Offset: 0xc | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
5:1	X	DEVEP: VUSB_HS_DV_EP: Number of endpoints supported by this device controller. Set to 16. This includes control endpoint 0.
0	X	DC: Device capable: Set to 1 indicating support for device mode.

19.13.1.4 USB2_CONTROLLER_USB2D_HW_TXBUF_0

USB2D Hardware TX Buffer Register

Offset: 0x10 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	TXCHANADD: VUSB_HS_TX_CHAN_ADD: Total number of address bits for the transmit buffer of each transmit endpoint. Set to 7. Each transmit buffer is 128 words deep.
15:8	X	TXADD: VUSB_HS_TX_ADD: Total number of address bits for the transmit buffer. Set to 11. The total depth of the transmit buffer is 2048 words.
7:0	X	TCBURST: VUSB_HS_TX_BURST: Maximum burst size supported by the transmit endpoints for data transfers. Set to 8.

19.13.1.5 USB2_CONTROLLER_USB2D_HW_RXBUF_0

USB2D RX Buffer HW Parameters Register

Offset: 0x14 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:8	X	RXADD: VUSB_HS_RX_ADD: Total number of address bits for the receive buffer. Set to 7. The total depth of the receive buffer is 128 words
7:0	X	RXBURST: VUSB_HS_RX_BURST: Maximum burst size supported by the receive endpoints for data transfers. Set to 8.

19.13.1.6 USB2_CONTROLLER_USB2D_GPTIMER0LD_0

The host/device controller drivers can measure time-related activities using these timer registers. These registers are not part of the standard EHCI controller.

Offset: 0x80 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
23:0	0x0	GPTIMER0LD: This field is the value to be loaded into the GPTCNT countdown timer on a reset action. The value in this register represents the time in microseconds minus 1 for the timer duration .

19.13.1.7 USB2_CONTROLLER_USB2D_GPTIMER0CTRL_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00XXXXXX (0b00xxxxx0xxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	0x0	GTPRUN: This bit enables the general-purpose timer to run. Setting or clearing this bit will not

Bit	R/W	Reset	Description
			have an effect on the GPTCNT counter value.
30	WO	0x0	GPTRST: Writing a one to this bit will reload the GPTCNT with the value in GPTLD.
24	RW	0x0	GPTMODE: This bit selects between a single timer countdown and a looped countdown. In one-shot mode, the timer will count down to zero, generate an interrupt, and stop until the counter is reset by software. In repeat mode, the timer will count down to zero, generate an interrupt, and automatically reload the counter to begin again.
23:0	RO	X	GPTCNT: This field is the value of the running timer.

19.13.1.8 USB2_CONTROLLER_USB2D_GPTIMER1LD_0

Offset: 0x88 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000)

Bit	Reset	Description
23:0	0x0	GPTIMER1LD: This field is the value to be loaded into the GPTCNT countdown timer on a reset action. The value in this register represents the time in microseconds minus 1 for the timer duration .

19.13.1.9 USB2_CONTROLLER_USB2D_GPTIMER1CTRL_0

Offset: 0x8c | Read/Write: R/W | Reset: 0x00XXXXXX (0b00xxxxx0xxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	0x0	GTPRUN: This bit enables the general-purpose timer to run. Setting or clearing this bit will not have an effect on the GPTCNT counter value.
30	WO	0x0	GPTRST: Writing a one to this bit will reload the GPTCNT with the value in GPTLD.
24	RW	0x0	GPTMODE: This bit selects between a single timer countdown and a looped countdown. In one-shot mode, the timer will count down to zero, generate an interrupt, and stop until the counter is reset by software. In repeat mode, the timer will count down to zero, generate an interrupt, and automatically reload the counter to begin again.
23:0	RO	X	GPTCNT: This field is the value of the running timer.

19.13.1.10 USB2_CONTROLLER_USB2D_CAPLENGTH_0

USB2D Capability Register Length Register

Offset: 0x100 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
7:0	X	CAPLENGTH: Indicates which offset to add to the register base address at the beginning of the Operational Register. Set to 0x30.

19.13.1.11 USB2_CONTROLLER_USB2D_HCIVERSON_0

USB2D Host Interface Version Number Register

Offset: 0x102 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	HCIVERSION: Contains a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision. This host controller supports EHCI revision 1.00.

19.13.1.12 USB2_CONTROLLER_USB2D_HCSPARAMS_0

USB2D Host Control Structural Parameters Register

Offset: 0x104 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
27:24	X	N_TT: Number of Transaction Translators: Indicates the number of embedded transaction translators associated with the USB2.0 host controller. This field is always set to 1 indicating only 1 embedded TT is implemented in this implementation. This is a non-EHCI field to support embedded TT.
23:20	X	N_PTT: Number of Ports per Transaction Translator: Indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. Field always equals N_PORTS. This is a non-EHCI field to support embedded TT.
15:12	X	N_CC: Number of Companion Controller: Indicates the number of companion controllers. This field is set to 0.
11:8	X	N_PCC: Number of Ports per Companion Controller: Indicates the number of ports supported per internal companion controller. This field is set to 0.
4	X	PPC: Port Power Control: Indicates whether the host controller implementation includes port power control. 1 = Ports have port power switches 0= Ports do not have port power switches. This field affects the functionality of the port Power field in each port status and control register. This field is set to 1.
3:0	X	N_PORTS: Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller. This field is fixed to 1, since this host controller only supports 1 port.

19.13.1.13 USB2_CONTROLLER_USB2D_HCCPARAMS_0

USB2D Host Control Capability Parameters Register

Offset: 0x108 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
18	X	PPC: Per-Port Change Event Capability - RO. Default = 1b. This field indicates the support for per-port change events. This field is related to the USBCMD PPE field, USBSTS PPCI field, and USBINTR PPCE field.
17	X	LEN: Link Power Management Capability - RO. Default = 1b. This field indicates the support for LPM L1 state. This field is related to USBCMD HIRD field, POSTSCx SSTS and DA fields and HOSTPCx LEN, BA, and EPLPM fields.
15:8	X	EECP: EHCI Extended Capabilities Pointer: Indicates a capabilities list exists. A value of 00h indicates no extended capabilities are implemented. For this implementation this field is always "0".
7:4	X	IST: Isochronous Scheduling Threshold. This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit [7] is zero, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit [7] is a one, then host software assumes the host controller may cache an isochronous data structure for an entire frame. This field will always be "0".
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".

19.13.1.14 USB2_CONTROLLER_USB2D_DCIVERSION_0

USB2D Device Interface Version Number Register

Offset: 0x120 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	DCIVERSION: The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.

19.13.1.15 USB2_CONTROLLER_USB2D_DCCPARAMS_0

USB2D Device Control Capabilities Register

Offset: 0x124 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
8	X	HC: Host Capable: 1 = This controller can operate as an EHCI compatible USB 2 0 host controller. This field is set to 1.
7	X	DC: Device Capable: 1 = Controller can operate as a USB 2.0 device. This field is set to 1.
5	X	LEN: Link Power Management Capability - RO. Default = 1b. This field indicates the support for LPM L1 state. This field is related to the DEVLCx ASUS, STL, BA, and NYT fields.
4:0	X	DEN: Device Endpoint Number: Number of endpoints built into the device controller. This is set to 16.

19.13.1.16 USB2_CONTROLLER_USB2D_EXTSTS_0

USB2D EXTSTS Register

Offset: 0x128 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000x

Bit	R/W	Reset	Description
4	RW	0x0	TI1: General Purpose Timer Interrupt 1 - RWC. Default = 0b. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero. Writing a one to this bit will clear it.
3	RW	0x0	TI0: General Purpose Timer Interrupt 0 - RWC. Default = 0b. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero. Writing a one to this bit will clear it.
2	RW	0x0	UPA: USB Host Periodic Interrupt (USBHSTPERINT) R/WC. This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set and the TD was from the periodic schedule. This bit is also set by the Host Controller when a short packet is detected AND the packet is on the periodic schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be zero. 0 = DISABLE 1 = ENABLE
1	RW	0x0	UAI: USB Host Asynchronous Interrupt (USBHSTASYNCINT) R/WC. This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set AND the TD was from the asynchronous schedule. This bit is also set by the Host when a short packet is detected AND the packet is on the asynchronous schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be zero. 0 = DISABLE 1 = ENABLE
0	RO	X	NAKI: NAK Interrupt Bit Read Only. This bit is read only. It is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and the corresponding TX/RX Endpoint NAK Enable bit are set. This bit is automatically cleared by hardware when the all the enabled TX/RX Endpoint NAK bits are cleared. 0 = DISABLE 1 = ENABLE

19.13.1.17 USB2_CONTROLLER_USB2D_USBEXTINTR_0

USB2D EXTINTR Register

Offset: 0x12c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000

Bit	Reset	Description
4	0x0	TIE1: General Purpose Timer Interrupt Enable 1 - RWC. Default = 0b. When this bit is a one, and the TI1 bit in the EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the TI1 bit.
3	0x0	TIE0: General Purpose Timer Interrupt Enable 0 - RWC. Default = 0b. When this bit is a one, and the TI1 bit in the EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the TI1 bit.
2	0x0	UPIE: UPIE Interrupt Enable. 1 = USB controller issues an interrupt if the UPA bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE
1	0x0	UAIE: UAIE Interrupt Enable. 1 = USB controller issues an interrupt if the UAI bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE
0	0x0	NAKE: NAK Interrupt Enable. 1 = USB controller issues an interrupt if the NAKI bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE

19.13.1.18 USB2_CONTROLLER_USB2D_USBCMD_0

USB2D USB Command Register

Offset: 0x130 | Read/Write: R/W | Reset: 0bxxxx00000000010000000x1x11x0000000

Bit	R/W	Reset	Description
27:24	RW	0x0	HIRD: Host Initiated Resume Duration RW. Default = 0000b. This has the same behavior as bits 7:4 of the BA field of the HOSTPCx register. When writing to this field all BA[7:4] fields of all HOSTPCx registers will be set to this value. This field is used by system software to specify the minimum amount of time the host controller will drive the K-state during a host-initiated resume from an LPM state (e.g., L1), and is conveyed to each LPM-enabled device (via the HIRD bits within an LPM Tokens bmAttributes field) upon entry into a low-power state. Note the host controller is required to drive resume signaling for at least the amount of time specified in the HIRD value conveyed to the device during any proceeding host-initiated resume. Also note that the host controller is not required to observe this requirement during device-initiated resumes. Encoding for this field is identical to the definition for the similarly named HIRD field within an LPM Token, specifically: a value 0000b equals 50 µs and each additional increment adds 75 µs. For example, the value 0001b equals 125 µs, and the value 1111b equals 1,175 µs (~1.2 ms).
23:16	RW	0x8	ITC: Interrupt Threshold Control .Read/Write. Default 08h. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are shown below. Value Maximum Interrupt Interval. 00h: Immediate (no threshold). 01h: 1 micro-frame. 02h: 2 micro-frames. 04h: 4 micro-frames. 08h: 8 micro-frames. 10h: 16 micro-frames. 20h: 32 micro-frames. 40h: 64 micro-frames 0 = IMMEDIATE 2 = ONE_MF 4 = TWO_MF 8 = EIGHT_MF 16 = SIXTEEN_MF

Bit	R/W	Reset	Description
			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 queue head (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 the Programmable Frame List Flag in the HCCPARAMS registers is set to one. Hence this field is Read/Write for this implementation. This field specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. Note that this field is made up from USBCMD bits 15, 3, and 2. 000 = 1024 elements (4096 bytes) Default value 001 = 512 elements (2048 bytes) 010 = 256 elements (1024 bytes) 011 = 128 elements (512 bytes)

Bit	R/W	Reset	Description
			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

19.13.1.19 USB2_CONTROLLER_USB2D_USBSTS_0

USB2D USB Status Register

Offset: 0x134 | Read/Write: R/W | Reset: 0b00000000000000000x100x0000x0000

Bit	R/W	Reset	Description
31:16	RW	0x0	PPCI: Port-n Change Detect - RW. Default = 0000h. The definition for each bit is identical to the Port Change Detect field (bit 2 of this register) except these bits are specific to a given port, where bit 16 = Port 1, 17 = Port 2, etc. For example, if bit 17 is set to a one then a port change event was detected on Port 2. The N_PORTS field in HCSPARAMS specifies how many ports are exposed by the host controller and thus how many bits in this field are valid.
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

Bit	R/W	Reset	Description
13	RO	X	RCL: Reclamation. This is a read-only status bit used to detect an empty asynchronous schedule. Only used by the host controller. 0 = DISABLE 1 = ENABLE
12	RW	0x1	HCH: HCHalted. 1 = Default. This bit is a zero whenever the Run/Stop bit is a one. The Host Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g., internal error). Only used by the host controller. 0 = UNHALTED 1 = HALTED
11	RW	0x0	UALT_INT: ULPI alt_int Interrupt. 0 = Default. This interrupt bit is set when an RXCMD is received through the ULPI interface with bit 7 set (alt_int).The alt_int itself is set when an unmasked event occurs on any bit in the Carkit Interrupt Latch Register, in the ULPI PHY The software should read the Carkit Interrupt Latch Register (Read to Clear) through the ULPI Viewport to check the source of the interrupt. Only present in designs where configuration constant VUSB_HS_PHY_ULPI = 1. 0 = NOT_ULPI_ALT_INT 1 = ULPI_ALT_INT
10	RW	0x0	ULPI_INT: ULPI Interrupt. This bit is set whenever an interrupt is received from ULPI PHY. Software writes 1 to clear it. 0 = NOT_ULPI_INT 1 = ULPI_INT
8	RW	0x0	SLI: DCSuspend. When a device controller enters a suspend state from an active state, this bit will be set to a 1. The device controller clears the bit upon exiting from a suspend state. Only used by the device controller. 0 = NOTSUSPEND 1 = SUSPENDED
7	RW	0x0	SRI: SOF Received. When the device controller detects a Start Of (micro) Frame, this bit will be set to a one. When an SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1 ms in device FS mode and every 125 µs in HS mode and will be synchronized to the actual SOF that is received. Since the device controller is initialized to FS before connect, this bit will be set at an interval of 1 ms during the prelude to the connect and chirp. In host mode, this bit will be set every 125 µs and can be used by the host controller driver as a time base. Software writes a 1 to this bit to clear it. This is a non-EHCI status bit. 0 = SOF_NOT_RCVD 1 = SOF_RCVD
6	RW	0x0	URI: USB Reset Received. When the device controller detects a USB Reset and enters the default state, this bit is set to a 1. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used by the device controller. 0 = NO_USB_RESET 1 = USB_RESET
5	RW	0x0	AAI: Interrupt and Asynchronous Advance. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the USBCMD register. This status bit 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 had its interrupt on complete (IOC) bit set. 0 = NO_ERROR 1 = ERROR
0	RW	0x0	UI: USB Interrupt. This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) as an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes. 0 = NO_INT 1 = INT

19.13.1.20 USB2_CONTROLLER_USB2D_USBINTR_0

USB2D USB Interrupt Enable Register

Offset: 0x138 | Read/Write: R/W | Reset: 0b0000000000000000xxxx00x000000000

Bit	Reset	Description
31:16	0x0	PPCE: Port-n Change Detect Enable - RW. Default = 0000h. The definition for each bit in this field is identical to bit 2 of this register (Port Change Interrupt Enable) except these bits are specific to a given port, where bit 16 = Port 1, 17 = Port 2, etc. For example, if bit 17 is set (1b) then a port change event was detected on Port 2. When a bit in this field is a one, and the corresponding Port-n Change Detect bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Port-n Change Detect bit.
11	0x0	UALTIE: ULPI alt_int Interrupt Enable. 1 = USB controller issues an interrupt if the ULPI_ALT_INT bit in the USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the ULPI_ALT_INT bit. 0 = DISABLE 1 = ENABLE
10	0x0	ULPIE: ULPI Interrupt Enable. 1 = USB controller issues an interrupt if ULPI_INT bit in USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the ULPI_INT bit. 0 = DISABLE 1 = ENABLE
8	0x0	SLE: Sleep Enable. 1 = Device controller issues an interrupt if DCSuspend bit in USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the DCSuspend bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE
7	0x0	SRE: SOF Received Enable. 1 = Device controller issues an interrupt if SOF Received bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the SOF Received bit. 0 = DISABLE 1 = ENABLE
6	0x0	URE: USB Reset Enable.1 = Device controller issues an interrupt if USB Reset Received bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the USB Reset Received bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	0x0	AAE: Interrupt on Asynchronous Advance Enable. 1 = The host controller issues an interrupt at the next interrupt threshold if Interrupt on Async Advance bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the Interrupt on Async Advance bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
4	0x0	SEE: System Error Enable. 1 = Host/device controller issues an interrupt if the System Error bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the System Error bit. 0 = DISABLE 1 = ENABLE
3	0x0	FRE: Frame List Rollover Enable. 1 = Host controller issues an interrupt if Frame List Rollover bit in the USBSTS register = 1. The interrupt is acknowledged by software clearing the Frame List Rollover bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
2	0x0	PCE: Port Change Detect Enable. 1 = Host/device controller issues an interrupt if Port Change Detect bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the Port Change Detect bit. 0 = DISABLE 1 = ENABLE
1	0x0	UEE: USB Error Interrupt Enable. 1 = Host controller issues an interrupt at the next interrupt threshold if the USBERRINT bit in USBSTS = 1. The interrupt is acknowledged by software clearing the USBERRINT bit in the USBSTS register. 0 = DISABLE 1 = ENABLE
0	0x0	UE: USB Interrupt Enable. 1 = Host/device issues an interrupt at the next interrupt threshold if the USBINT bit in USBSTS = 1. The interrupt is acknowledged by software clearing the USBINT bit. 0 = DISABLE 1 = ENABLE

19.13.1.21 USB2_CONTROLLER_USB2D_FRINDEX_0

USB2D USB Frame Index Register

Offset: 0x13c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
13:0	X	FRINDEX: Frame Index. The value in this register increments at the end of each time frame (micro-frame). Bits [N: 3] are used for the Frame List current index. Each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in host mode. USBCMD [Frame List Size] Number Elements N 000b (1024) 12 001b (512) 11 010b (256) 10 011b (128) 9 100b (64) 8 101b (32) 7 110b (16) 6 111b (8) 5 In device mode, the value is the current frame number of the last frame transmitted. It is not used as an index. In either mode bits 2:0 indicate the current micro-frame.

19.13.1.22 USB2_CONTROLLER_USB2D_PERIODICLISTBASE_0

USB2D Host Controller Frame List Base Address Register

Offset: 0x144 | Read/Write: R/W | Reset: 0b00000000000000000000xxxxxxxxxxxx

Bit	Reset	Description
31:25	0x0	USBADR: Device mode. The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS request.
24	0x0	USBADRA: Device Address Advance. Default=0. When this bit is 0, any writes to USBADR are instantaneous. When this bit is written to a 1 at the same time or before USBADR is written, the write to the USBADR field is staged and held in a hidden register. After an IN occurs on endpoint 0 and is ACKed, USBADR will be loaded from the holding register. Hardware will automatically clear this bit on the following conditions: 1) IN is ACKed to endpoint 0. (USBADR is updated from staging register). 2) OUT/SETUP occur to endpoint 0. (USBADR is not updated). 3) Device Reset occurs (USBADR is reset to 0). Note: After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field. This mechanism will ensure this specification is met when the DCD cannot write of the device address within 2 ms from the SET_ADDRESS status phase. If the DCD writes the USBADR with USBADRA=1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR will be programmed instantly at the correct time and meet the 2 ms USB requirement.
31:12	0x0	BASEADR: Host mode: This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. The HCD loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register are combined with the Frame Index Register (FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence. Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.

19.13.1.23 USB2_CONTROLLER_USB2D_ASYNCLISTADDR_0

USB2D Next Asynchronous List Address Register

Offset: 0x148 | Read/Write: R/W | Reset: 0b00000000000000000000000000xxxxx

Bit	Reset	Description
31:11	0x0	EPBASE: Device mode. This register contains the address of the top of the endpoint list in system memory. These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Heads (QHs). 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.

19.13.1.24 USB2_CONTROLLER_USB2D_ASYNCTTSTS_0

USB2D Asynchronous Buffer Status for Embedded TT Register

Offset: 0x14c | Read/Write: R/W | Reset: 0bx0000000xxxxxxxxxxxxxxxxxxxxxx0x

Bit	R/W	Reset	Description
30:24	RW	0x0	TTHA: Internal TT Hub Address representation. This field is used to match the Hub Address field in Queue Head (QH) and split isochronous transaction descriptor (siTD) to determine if the packet is routed to the internal transaction translator (TT) for directly attached FS/LS devices. If the Hub Address in the QH or siTD does not match this address then the packet will be broadcast on the High Speed ports destined for a downstream High Speed hub with the address in QH/siTD.
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

Bit	R/W	Reset	Description
			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.

19.13.1.25 USB2_CONTROLLER_USB2D_BURSTSIZE_0

USB2D Burst Size Register

Offset: 0x150 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000100000001000

Bit	Reset	Description
15:8	0x8	TXPBURST: Programmable TX Burst Length. (Read/Write). This register represents the maximum length of a burst in 32-bit words while moving data from system memory to the USB bus.
7:0	0x8	RXPBURST: Programmable RX Burst Length. (Read/Write). This register represents the maximum length of a burst in 32-bit words while moving data from the USB bus to system memory.

19.13.1.26 USB2_CONTROLLER_USB2D_TXFILLTUNING_0

USB2D Transmit Fill Tuning Register

Offset: 0x154 | Read/Write: R/W | Reset: 0bxxxxxxxxxx000010xxx0000000000000

Bit	Reset	Description
21:16	0x2	TXFIFOTHRES: FIFO Burst Threshold. (Read/Write). This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USBMODE register is set.
12:8	0x0	TXSCHHEALTH: Scheduler Health Counter. (Read/Write To Clear) [Default = 0] This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max out at 31.
7:0	0x0	TXSCHOH: Scheduler Overhead. (Read/Write) [Default = 0] This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267 µs when a device is connected in High-Speed Mode. The time unit represented in this register is 6.333 µs when a device is connected in Low/Full Speed Mode

19.13.1.27 USB2_CONTROLLER_USB2D_ICUSB_CTRL_0

This register enables and controls the ICUSB FS/LS transceiver.

USB2D ICUSB control register

Offset: 0x15c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000

Bit	Reset	Description
3	0x0	IC_ENB1: ICUSB transceiver enable. This bit enables the ICUSB transceiver. To enable the interface, the bits PTS must be set to 11 in the PORTSCx. Writing a '1' to this bit selects the IC_USB interface. 0 = DISABLE

Bit	Reset	Description
		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.

19.13.1.28 USB2_CONTROLLER_USB2D_ULPI_VIEWPORT_0

This register provides indirect access to the ULPI PHY register set. Although the USB controller performs access to the ULPI PHY register set, there may be extraordinary circumstances where software may need direct access.

Note: WRITES TO THE ULPI THROUGH THE VIEWPORT CAN SUBSTANTIALLY HARM STANDARD USB OPERATIONS. CURRENTLY NO USAGE MODEL HAS BEEN DEFINED WHERE SOFTWARE SHOULD NEED TO EXECUTE WRITES DIRECTLY TO THE ULPI PHY. SEE EXCEPTION REGARDING OPTIONAL FEATURES BELOW.

Note: Executing read operations through the ULPI Viewport should have no harmful side effects to standard USB operations.

There are two operations that can be performed with the ULPI Viewport: wakeup and read /write operations.

The wakeup operation is used to put the ULPI interface into normal operation mode and reenable the clock if necessary. A wakeup operation is required before accessing the registers when the ULPI interface is operating in low power mode, serial mode, or carkit mode.

The ULPI state can be determined by reading the sync state bit (ULPI_SYNC_STATE). If this bit is a one, then ULPI interface is running in normal operation mode and can accept read/write operations. If the ULPI_SYNC_STATE indicates a 0 then then read/write operations will not be able to execute. Undefined behavior will result if ULPI_SYNC_STATE = 0 and a read or write operation is performed.

To execute a wakeup operation, write all 32-bits of the ULPI Viewport where ULPI_PORT is constructed appropriately and the ULPI_WAKEUP bit is a 1 and ULPI_RUN bit is a 0. Poll the ULPI Viewport until ULPI_WAKEUP is zero for the operation to complete.

To execute a read or write operation, write all 32 bits of the ULPI Viewport where ULPI_DATA_WR, ULPI_REG_ADDR, ULPI_PORT, ULPI_RD_WR are constructed appropriately and the ULPI_RUN bit is a 1. Poll the ULPI Viewport until ULPI_RUN is zero for the operation to complete. Once ULPI_RUN is zero, the ULPI_DATA_RD will be valid if the operation was a read.

The polling method above can be changed to interrupt driven using the ULPI interrupt defined in the USBSTS and USBINTR registers. When a wakeup or read/write operation is complete, the ULPI_INT interrupt will be set.

There are several optional features that may need to be enabled or disabled by system software as part of system configuration. These bits are contained in the Interface and OTG Control registers of the ULPI PHY register set. These registers also contain bits which are controlled by the link dynamically and therefore should be only modified by system software using the Set/Clear access method. Direct writes to these registers could have harmful side effects to the standard USB operations. The optional bits are as follows: Bits 3 through 7 in the Interface Control register and Bits 6 and 7 in the OTG Control register.

Please refer to the ULPI Specification Revision 1.1 for further information on the use of the optional features.

USB2D ULPI Viewport Register

Offset: 0x160 | Read/Write: R/W | Reset: 0b000xx00000000000xxxxxxxx00000000

Bit	R/W	Reset	Description
31	RW	0x0	ULPI_WAKEUP: ULPI Wakeup. Writing the 1 to this bit will begin the wakeup operation. The bit will automatically transition to 0 after the wakeup is complete. Once this bit is set, the driver cannot set it back to 0. Note: The driver must never execute a wakeup and a read/write operation at the same time. 0 = CLEAR 1 = SET
30	RW	0x0	ULPI_RUN: ULPI read/write Run. Writing the 1 to this bit will begin the read/write operation. The bit will automatically transition to 0 after the read/write is complete. Once this bit is set, the driver cannot set it back to 0. Note: The driver must never execute a wakeup and a read/write operation at the same time. 0 = CLEAR 1 = SET
29	RW	0x0	ULPI_RD_WR: ULPI read/write control. (0) Read; (1) Write. This bit selects between running a read or write operation. 0 = READ 1 = WRITE
27	RO	X	ULPI_SYNC_STATE: ULPI sync state. (1) Normal Sync. State. (0) In another state (i.e., carkit, serial, low power). This bit represents the state of the ULPI interface. 0 = NOT_NORMAL 1 = NORMAL
26:24	RW	0x0	ULPI_PORT: ULPI PHY port number. This field should be always written as 0.
23:16	RW	0x0	ULPI_REG_ADDR: ULPI PHY register address. When doing a read or write operation to the ULPI PHY, the address of the ULPI PHY register being accessed is written to this field.
15:8	RO	X	ULPI_DATA_RD: ULPI PHY data read. The data from the ULPI PHY register can be read from here after the read operation completes.
7:0	RW	0x0	ULPI_DATA_WR: ULPI PHY data write. The data to write to the ULPI PHY register is written here.

19.13.1.29 USB2_CONTROLLER_USB2D_PORTSC1_0

USB2D Port Status/Control 1 Register

Offset: 0x174 | Read/Write: R/W | Reset: 0b0000000xx0000000xxx1xx00x0xx010x

Bit	R/W	Reset	Description
31:25	RW	0x0	DA: Device Address - RW. Default = 0000000b. The 7-bit USB device address for the device attached to an immediately downstream of the associated root port. A value of zero indicates no device is present or software support for this feature is not present. This is used by the Controller when sending the LPM token. This field is only valid when the core is operating in host mode. If in device mode it will be read only and always equal to 0000000b.
24:23	RO	X	SSTS: Suspend Status - RO. Default = 00b. These two bits are used by software to determine whether an L1-based suspend request was successful, specifically: 00b - L1 state entered with success. ACK received from peripheral. 01b - NYET received from peripheral. It was not able to enter L1 state this time. 10b - L1 state not supported by peripheral. STALL received. 11b - Peripheral did not respond or an error occurred. The value of this field is only valid when the port resides in the L0 state - that is, the meaning of these bits is invalid whenever bit 7 of this register (SUSP) is one. Ideally, the Controller driver should read this register if it receives an interrupt after issuing a suspend using L1 support. In case of a non-success a port change interrupt will be fired and this field should be checked for a possible L1 failure. This field is only valid when the core is operating in host mode. If in device mode, it will be always equal to 00b. 0 = L1STATE_ENTERED 1 = NYET_PERIPH 2 = L1STATE_NOT_SUPPORTED

Bit	R/W	Reset	Description
			3 = PERIPH_NORESP_ERR
22	RW	0x0	WKOC: Default = 0b. Wake on Over-current Enable: Writing this bit to a one enables the port to be sensitive to over-current conditions as wake-up events. This field is zero if Port Power (PP) is zero. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined behavior. 0 = DISABLE 1 = ENABLE
21	RW	0x0	WKDS: Wake on Disconnect Enable: Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This field is zero if Port Power (PP) is zero or in device mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined behavior. This bit should not be written to 1 if there is no device connected. After the device disconnect is detected, this bit should be cleared to 0. 0 = DISABLE 1 = ENABLE
20	RW	0x0	WKCN: Wake on Connect Enable: Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This field is zero if Port Power (PP) is zero or in device mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined behavior. This bit should not be written to 1 while the device is connected. After the device connection is detected, this bit should be cleared to 0. 0 = DISABLE 1 = ENABLE
19:16	RW	0x0	PTC: Port Test Control: Any value other than zero indicates that the port is operating in test mode. Value Specific Test. 0000b: Not enabled. 0001b: J_ STATE. 0010b: K_STATE. 0011b: SEQ_NAK. 0100b: Packet. 0101b: FORCE_ENABLE. 0110b to 1111b: Reserved. | Refer to Chapter 7 of the USB Specification Revision 2.0 for details on each test mode. 0 = NORMAL_OP 1 = TEST_J 2 = TEST_K 3 = TEST_SE0_NAK 4 = TEST_PKT 5 = TEST_FORCE_ENABLE
15:14	RO	X	PIC: Port Indicator Control: This field is not supported in the current implementation. Please use a GPIO if you wish to use Port Indicators.
13	RO	X	PO: Port Owner. Port owner handoff is not implemented in this design, therefore this bit will always be 0.
12	RW	0x1	PP: Port Power: The function of this bit depends on the value of the Port Power Switching (PPC) field in the HCSPARAMS register. The behavior is as follows: PPC PP Operation 0b 0b Read Only. A device controller with no OTG capability does not have port power control switches. 1b 1b/0b RW. Host/OTG controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (i.e., PP equals a 0), the port is non-functional and will not report attaches, detaches, etc. When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitioned by the host controller driver from a one to a zero (removing power from the port). 0 = NOT_POWERED 1 = POWERED

Bit	R/W	Reset	Description
11:10	RO	X	LS: These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. The encoding of the bits is: 00b = SE0. 01b = K-state. 10b = L-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 = K_STATE 2 = J_STATE 3 = UNDEFINED
9	RW	0x0	SLP: Suspend using L1 - RW. Default = 0b. When this bit is set to '1' and a non-zero Device Address (DA) is specified, the Controller will instigate L1 entry during suspend (bit 7) and L1 exit during resume (bit 6). When this bit is set to zero, the Controller will use the legacy (L2) mechanism. Software should only set this bit when the device attached immediately downstream supports L1 transitions. When acting as device, this bit is read-only and set to '1' by the hardware when the Controller enters is L1 state (LPM token received and accepted). Note: HSP is redundant with PSPD[27:26]. This bit is not defined in the EHCI specification.
8	RW	0x0	PR: This field is zero if Port Power (PP) is zero. In Host Mode: Read/Write. 1=Port is in Reset. 0=Port is not in Reset. When software writes a one to this bit the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver. In Device Mode: This bit is a read only status bit. Device reset from the USB bus is also indicated in the USBSTS register. 0 = NOT_USB_RESET 1 = USB_RESET
7	RO	X	SUSP: Port suspend. 1=Port in suspend state. 0=Port not in suspend state. In Host Mode: Read/Write. Port Enabled Bit and Suspend bit of this register define the port states as follows: Bits [Port Enabled, Suspend] Port State 0x Disable 10 Enable 11 Suspend. When in the suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to zero when software sets the Force Port Resume bit to zero. A write of zero to this bit is ignored by the host controller. If host software sets this bit to a one when the port is not enabled (i.e., Port enabled bit is a zero), the results are undefined. This field is zero if Port Power (PP) is zero in host mode. In Device Mode: Read Only. This bit is a read only status bit. 0 = NOT_SUSPEND 1 = SUSPEND

Bit	R/W	Reset	Description
6	RW	0x0	FPR: Force Port Resume. 1= Resume detected/driven on port. 0=No resume (K state) detected/driven on port. In Host Mode: Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver. Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. This bit remains a one until the port has switched to the high-speed idle. Writing a zero has no effect because the port controller will time the resume operation to clear the bit when the port control state switches to HS or FS idle. This field is zero if Port Power (PP) is zero in host mode. This bit is not-EHCI compatible. In Device mode: After the device has been in Suspend State for 5ms or more, software must set this bit to one to drive resume signaling before clearing. The Device Controller will set this bit to one if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. Software should ensure that the PHY clock is operational before writing a 1 to this bit to start the resume sequence. This is true for both Device and Host modes. 0 = NO_RESUME 1 = RESUME
5	RO	X	OCC: Over-current Change: Not supported 0 = NO_CHANGE 1 = CHANGE
4	RO	X	OCA: Over-current Active: Not supported 0 = NO_OVER_CURRENT 1 = OVER_CURRENT
3	RW	0x0	PEC: Port Enable/Disable Change: 1=Port enabled/disabled status has changed. 0=No change. In Host Mode: For the root hub, this bit gets set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a one to it. This field is zero if Port Power (PP) is zero. In Device mode: The device port is always enabled. (This bit will be zero.) 0 = NO_CHANGE 1 = CHANGE
2	RW	0x1	PE: Port Enabled/Disabled: 1=Enable. 0=Disable (default). In Host Mode: Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0b) downstream propagation of data is blocked except for reset. This field is zero if Port Power (PP) is zero in host mode. In Device Mode: The device port is always enabled. (This bit will be one) 0 = PORT_DISABLED 1 = PORT_ENABLED
1	RW	0x0	CSC: Connect Status Change: 1 =Change in Current Connect Status. 0=No change (default) In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (i.e., the bit will remain set). Software clears this bit by writing a one to it. This field is zero if Port Power (PP) is zero in host mode. This bit is undefined in device controller mode. 0 = NO_CHANGE 1 = CHANGE

Bit	R/W	Reset	Description
0	RO	X	CCS: Current Connect Status. In Host Mode: 1=Device is present on port. 0=No device is present (default). This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set. This field is zero if Port Power (PP) is zero in host mode. In Device Mode: 1=Attached 0=Not Attached (default) A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended. 0 = NOT_CONNECTED 1 = CONNECTED

19.13.1.30 USB2_CONTROLLER_USB2D_HOSTPC1_DEVLC_0

USB2D Device Mode LPM Behavior and Control Register

This register controls the LPM behavior and controls the behavior of each USB port. Only available in device mode. This register shares the same address with the HOSTPC1 register (which is used only in host mode).

USB2D Host Mode LPM Behavior and Control Register

This register controls the LPM behavior and controls the behavior of each USB port. Only available in host mode. This register shares the same address with the DEVLC register (which is used only in device mode).

Offset: 0x1b4 | Read/Write: R/W | Reset: 0b0000xxx0000000000000xxxxxxxxxxx0

Bit	R/W	Reset	Description
31:29	RW	0x0	PTS: Parallel transceiver select. This bit is not defined in the EHCI specification. 0 = UTMI 1 = RESERVED 2 = ULPI 3 = ICUSB_SER 4 = HSIC
28	RW	0x0	STS: Serial transceiver not selected. This is the only value supported. This bit is not defined in the EHCI specification. 0 = PARALLEL_IF 1 = SERIAL_IF
27	RO	X	PTW: Parallel Transceiver Width. Fixed to 0. This bit is not defined in the EHCI specification. 0 = EIGHT_BIT 1 = RESERVED
26:25	RO	X	PSPD: This register field indicates the speed at which the port is operating. 00 = Full Speed 01 = Low Speed 10 = High Speed. This bit is not defined in the EHCI specification. 0 = FULL_SPEED 1 = LOW_SPEED 2 = HIGH_SPEED 3 = RESERVED
24	RW	0x0	ALPD: Auto Low Power While Disconnect - RW. Default = 0b. If set, this feature will be enabled, and every time the port enters the disconnect state it will also enter low power state,disabling the transceiver clock. The behavior will be same as if the PHCD bit was enabled (in fact this bit will be set to 1 as soon as low power mode is enabled).When this field is set the WKCN field of PORTSCx register (Wake on Connect Enable) will also be set. This way the core will wake up in case a connect is detected. There will be a delay between the detection of a disconnect and actually enter in low power mode. This delay can be controlled by writing to the ALPDD field in the register USBMODE. 0 = DONT_AUTO_LOW_POWER_WHILE_DISCONNECT 1 = AUTO_LOW_POWER_WHILE_DISCONNECT

Bit	R/W	Reset	Description
23	RW	0x0	PFSC: Port Force Full Speed Connect - RW. Default = 0b. Writing this bit to a '1b' will force the port to only connect at Full Speed. It disables the chirp sequence that allows the port to identify itself as High Speed. This is useful for testing FS configurations with an HS host, hub or device. This bit is not defined in the EHCI specification. 0 = DONT_FORCE_FULL_SPEED 1 = FORCE_FULL_SPEED
22	RW	0x0	PHCD: PHY Low Power Suspend - Clock disable: Writing this bit to a 1 will disable the PHY clock. Write a 0 enables it. Reading this bit will indicate the status of the PHY clock. NOTE: The PHY clock cannot be disabled if it is being used as the system clock. In device mode, the PHY can be put into Low Power Clock Disable when the device is not running (USBCMD RS=0b) or the host has signaled suspend (PORTSCx SUSP=1b). Low Power Clock Disable will be cleared automatically when the host has signaled resume. Before forcing a resume from the device, the Controller driver must clear this bit. In host mode, the PHY can be put into Low Power Suspend Clock Disable when the downstream device has been put into suspend mode or when no downstream device is connected. Low Power Clock Disable is completely under the control of software. 0 = DISABLE 1 = ENABLE
21:20	RW	0x0	LPMX: Auto LPM set - RW. Default = 00b. This bit field is **valid during Host Mode Only**. For Device Mode, this bit is reserved. Value Meaning 00b Disables auto LPM. 01b If there is no activity for a certain number of SOFs the controller will send an LPM token, enter in suspend and issue a port change interrupt. 10b Same as above but without issuing an interrupt. 11b Reserved for futures LPM enhancements. The detection of no activity is based on the absence of response from the downstream device. Because of this limitation this feature should not be used if ISO OUT endpoints are being used (as no handshake is expected).
17	RW	0x0	ASUS: Auto Low Power - RW. Default = 0b. This bit field is **valid during Device Mode Only**. In Host Mode, it is part of the ELPM field. This bit is used to control the auto low power feature. If set, the auto low power feature will be enabled and every time the port enters in suspend state it will also enter in low power state, disabling the transceiver clock. The behavior will be the same as if the PHCD bit was enabled (in fact this bit will be set to '1' as soon as low power mode is enabled). 0 = DISABLE 1 = ENABLE
19:16	RW	0x0	EPLPM: Endpoint for LPM token - RW. Default = 0000b. This bit field is **valid during Host Mode Only**. For Device Mode, bits [19:18] are reserved and bit 17 is ASUS, while bit 16 is STL. This sets the endpoint number to which the LPM token will be sent. It will be directly mapped to the ENDP field in the LPM token with the EXT PID.
16	RW	0x0	STL: STALL reply to LPM token - RW. Default = 0b. This bit field is **valid during Device Mode Only**. In Host Mode, it is part of the ELPM field. When this bit is set to '1', the Controller will reply always with STALL to all incoming LPM tokens. This bit overrides the LPM NYET bit (NYT). 0 = DISABLE 1 = ENABLE
15:12	RW	0x0	LPMFRM: Auto LPM SOF Threshold - RW. Default = 0000b. This bit field is **valid during Host Mode Only**. For Device Mode, this field is reserved. This holds the SOF counter threshold. When the number of SOFs with no activity in between reaches this threshold and the auto LPM is enabled, an LPM token will be sent and the port will enter in suspend state. The SOF counter for this threshold is incremented each 125 µs, even if the port is not in HS operation.
11:1	RO	X	BA: bmAttributes - RO. Default = 00000000000b. This holds the bmAttributes field of the LPM sub-token received, after the EXT PID token.

Bit	R/W	Reset	Description
0	RW	0x0	NYT_ASUS: NYET reply to LPM token - RW. Default = 0b. This bit is NYT during Device Mode. Details follow: When this bit is to '1', the device controller will NYET all the LPM tokens. When this bit is set to '0', the Controller will ACK all the LPM tokens if the STALL bit (STL) is also set to '0'. This bit is ASUS in Host Mode. Details follow: This bit is used to control the auto low power feature. If set, the auto low power feature will be enabled and every time the port enters in suspend state it will also enter in low power state, disabling the transceiver clock. The behavior will be the same as if the PHCD bit was enabled (in fact this bit will be set to '1' as soon as low power mode is enabled). 0 = DISABLE 1 = ENABLE

19.13.1.31 USB2_CONTROLLER_USB2D_OTGSC_0

USB2D On-The-Go (OTG) Status and Control Register

Offset: 0x1f4 | Read/Write: R/W | Reset: 0bx0000000x0000000xxxxxxxxxx100x00

Bit	R/W	Reset	Description
30	RW	0x0	DPIE: Data Pulse Interrupt Enable. Setting this bit enables the Data pulse interrupt. 0 = DISABLE 1 = ENABLE
29	RW	0x0	ONEMSE: 1 millisecond timer Interrupt enable. Setting this bit enables the 1 millisecond timer interrupt. 0 = DISABLE 1 = ENABLE
28	RW	0x0	BSEIE: B Session End Interrupt Enable. Setting this bit enables the B session end interrupt 0 = DISABLE 1 = ENABLE
27	RW	0x0	BSVIE: B Session Valid Interrupt Enable. Setting this bit enables the B session valid interrupt 0 = DISABLE 1 = ENABLE
26	RW	0x0	ASVIE: A Session Valid Interrupt Enable. Setting this bit enables the A session valid interrupt 0 = DISABLE 1 = ENABLE
25	RW	0x0	AVVIE: A VBus Valid Interrupt Enable. Setting this bit enables the A VBus valid interrupt 0 = DISABLE 1 = ENABLE
24	RW	0x0	IDIE: USB ID Interrupt Enable. Setting this bit enables the USB ID interrupt 0 = DISABLE 1 = ENABLE
22	RW	0x0	DPIS: Data Pulse Interrupt Status. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USBMODE.CM = Host (11) and PORTSC(0). PortPower = Off (0). Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
21	RW	0x0	ONEMSS: 1 millisecond timer Interrupt Status: This bit is set once every millisecond. Software writes a 1 to clear it. 0 = INT_CLEAR 1 = INT_SET
20	RW	0x0	BSEIS: B Session End Interrupt Status. This bit is set when VBus has fallen below the B session end threshold. Software writes a 1 to clear this bit . 0 = INT_CLEAR 1 = INT_SET
19	RW	0x0	BSVIS: B Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the B session valid threshold (0.8 VDC). Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET

Bit	R/W	Reset	Description
18	RW	0x0	ASVIS: A Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the A session valid threshold (0.8 VDC). Software writes a one to clear this bit. 0 = INT_CLEAR 1 = INT_SET
17	RW	0x0	AVVIS: A VBus Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the VBus valid threshold (4.4 VDC) on an A device. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
16	RW	0x0	IDIS: USB ID Interrupt Status. This bit is set when a change on the ID input has been detected. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
14	RO	X	DPS: Data Bus Pulsing Status. A 1 indicates data bus pulsing is being detected on the port. 0 = STS_CLEAR 1 = STS_SET
13	RO	X	ONEMST: 1 millisecond timer toggle. This bit toggles once per millisecond 0 = STS_CLEAR 1 = STS_SET
12	RO	X	BSE: B session End. Indicates VBus is below the B session end threshold 0 = STS_CLEAR 1 = STS_SET
11	RO	X	BSV: B Session Valid. Indicates VBus is above the B session valid threshold 0 = STS_CLEAR 1 = STS_SET
10	RO	X	ASV: A Session Valid. Indicates VBus is above the A session valid threshold 0 = STS_CLEAR 1 = STS_SET
9	RO	X	AVV: A VBus Valid. Indicates VBus is above the A VBus valid threshold 0 = STS_CLEAR 1 = STS_SET
8	RO	X	ID: USB ID: 0 = A-device 1 = B-device 0 = A_DEV 1 = B_DEV
5	RW	0x1	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

19.13.1.32 USB2_CONTROLLER_USB2D_USBMODE_0

USB2D USB Device Mode Register

Offset: 0x1f8 | Read/Write: R/W | Reset: 0b00000000000000000xxxxxxxxx0xxxxx

Bit	R/W	Reset	Description
31:16	RW	0x0	ALPDD: Auto Low Power While Disconnect Delay, Used when the ALPD field of the HOSTPCx register is set. Defines the delay between disconnect detection and entering in low power mode. The delay is <this register value>*64*100/3 in milliseconds. The maximum value is 65535*64*100/3 = 139,808 ms. The minimum value is 0. Only used in host mode.
15	RW	0x0	SRT: Shorten USB Reset Time. Software should never set this to 1.
5	RW	0x0	VBPS: VBUS Power Select This can be used by logic that selects between an on-chip Vbus power source (charge pump) and an off-chip source in systems when both are available. Only to be used in Host Mode. No functionality is implemented for this, so software should not use this bit.
4	RO	X	SDIS: Stream disable: 1: Streaming is disabled - helpful to avoid overruns/underruns when the system load is too high. 0 = STREAM_ENABLE 1 = STREAM_DISABLE
3	RO	X	SLOM: Setup Lockout Mode: In device mode, this bit controls the behavior of the setup lockout mechanism. 0: Setup lockout is ON (default). 1: Setup lockout is OFF. Firmware requires the use of setup tripwire semaphore in USB2D_USBCMD register. 0 = LOCKOUT_ON 1 = LOCKOUT_OFF
2	RO	X	ES: Endian Select: Note: For this implementation, this bit should be always set to 0 (little endian). 0 = LITTLE_ENDIAN 1 = RESERVED
1:0	RO	X	CM: Controller Mode: The controller mode will default to an idle state and will need to be initialized to the desired operating mode after reset. This register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the RESET bit in the USBCMD register before reprogramming this register. 00 = Idle [Default]. 01 = Reserved. 10 = Device Controller. 11 = Host Controller. 0 = IDLE 1 = RESERVED 2 = DEVICE_MODE 3 = HOST_MODE

19.13.1.33 USB2_CONTROLLER_USB2D_ENDPTNAK_0

USB2D Endpoint NAK register

Offset: 0x200 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	EPTN: TX Endpoint NAK R/WC. Each TX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received IN token for the corresponding endpoint. 0 = CLEAR 1 = SET
15:0	0x0	EPRN: RX Endpoint NAK R/WC. Each RX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received OUT or PING token for the corresponding endpoint. 0 = CLEAR 1 = SET

19.13.1.34 USB2_CONTROLLER_USB2D_ENDPTNAK_ENABLE_0

USB2D Endpoint NAK Enable register

Offset: 0x204 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	EPTNE: TX Endpoint NAK Enable R/W. Each bit is an enable bit for the corresponding TX Endpoint NAK bit. If this bit is set and the corresponding TX Endpoint NAK bit is set, the NAK Interrupt bit is set. 0 = DISABLE 1 = ENABLE
15:0	0x0	EPRNE: RX Endpoint NAK Enable R/W. Each bit is an enable bit for the corresponding RX Endpoint NAK bit. If this bit is set and the corresponding RX Endpoint NAK bit is set, the NAK Interrupt bit is set. 0 = DISABLE 1 = ENABLE

19.13.1.35 USB2_CONTROLLER_USB2D_ENDPTSETUPSTAT_0

USB2D Endpoint Setup Status Register

Offset: 0x208 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
15	0x0	ENDPTSETUPSTAT15: Endpoint 15 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
14	0x0	ENDPTSETUPSTAT14: Endpoint 14 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
13	0x0	ENDPTSETUPSTAT13: Endpoint 13 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
12	0x0	ENDPTSETUPSTAT12: Endpoint 12 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
11	0x0	ENDPTSETUPSTAT11: Endpoint 11 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD

Bit	Reset	Description
10	0x0	ENDPTSETUPSTAT10: Endpoint 10 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
9	0x0	ENDPTSETUPSTAT9: Endpoint 9 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
8	0x0	ENDPTSETUPSTAT8: Endpoint 8 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
7	0x0	ENDPTSETUPSTAT7: Endpoint 7 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
6	0x0	ENDPTSETUPSTAT6: Endpoint 6 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
5	0x0	ENDPTSETUPSTAT5: Endpoint 5 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
4	0x0	ENDPTSETUPSTAT4: Endpoint 4 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
3	0x0	ENDPTSETUPSTAT3: Endpoint 3 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD

Bit	Reset	Description
2	0x0	ENDPTSETUPSTAT2: Endpoint 2 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
1	0x0	ENDPTSETUPSTAT1: Endpoint 1 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
0	0x0	ENDPTSETUPSTAT0: Endpoint 0 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD

19.13.1.36 USB2_CONTROLLER_USB2D_ENDPTPRIME_0

USB2D Endpoint Initialization Register

Offset: 0x20c | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	PETB15: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
30	0x0	PETB14: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
29	0x0	PETB13: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
28	0x0	PETB12: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
27	0x0	PETB11: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
26	0x0	PETB10: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
25	0x0	PETB9: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
24	0x0	PETB8: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
23	0x0	PETB7: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
22	0x0	PETB6: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
21	0x0	PETB5: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
20	0x0	PETB4: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
19	0x0	PETB3: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
18	0x0	PETB2: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
17	0x0	PETB1: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
16	0x0	PETB0: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
15	0x0	PERB15: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
14	0x0	PERB14: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
13	0x0	PERB13: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
12	0x0	PERB12: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
11	0x0	PERB11: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
10	0x0	PERB10: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
9	0x0	PERB9: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
8	0x0	PERB8: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
7	0x0	PERB7: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
6	0x0	PERB6: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
5	0x0	PERB5: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
4	0x0	PERB4: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
3	0x0	PERB3: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
2	0x0	PERB2: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
1	0x0	PERB1: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
0	0x0	PERB0: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

19.13.1.37 USB2_CONTROLLER_USB2D_ENDPTFLUSH_0

USB2D Endpoint De-Initialization Register

Offset: 0x210 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	FETB15: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
30	0x0	FETB14: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
29	0x0	FETB13: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
28	0x0	FETB12: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
27	0x0	FETB11: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
26	0x0	FETB10: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
25	0x0	FETB9: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
24	0x0	FETB8: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
23	0x0	FETB7: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
22	0x0	FETB6: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
21	0x0	FETB5: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
20	0x0	FETB4: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
19	0x0	FETB3: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
18	0x0	FETB2: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
17	0x0	FETB1: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
16	0x0	FETB0: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
15	0x0	FERB15: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
14	0x0	FERB14: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
13	0x0	FERB13: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
12	0x0	FERB12: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
11	0x0	FERB11: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
10	0x0	FERB10: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
9	0x0	FERB9: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
8	0x0	FERB8: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
7	0x0	FERB7: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
6	0x0	FERB6: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
5	0x0	FERB5: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
4	0x0	FERB4: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
3	0x0	FERB3: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
2	0x0	FERB2: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
1	0x0	FERB1: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
0	0x0	FERB0: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

19.13.1.38 USB2_CONTROLLER_USB2D_ENDPTSTATUS_0

USB2D Endpoint Status Register

Offset: 0x214 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	ETBR15: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
30	X	ETBR14: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
29	X	ETBR13: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
28	X	ETBR12: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
27	X	ETBR11: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
26	X	ETBR10: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
25	X	ETBR9: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
24	X	ETBR8: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
23	X	ETBR7: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
22	X	ETBR6: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
21	X	ETBR5: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
20	X	ETBR4: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
19	X	ETBR3: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
18	X	ETBR2: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
17	X	ETBR1: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
16	X	ETBR0: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
15	X	ERBR15: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
14	X	ERBR14: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
13	X	ERBR13: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
12	X	ERBR12: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
11	X	ERBR11: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
10	X	ERBR10: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
9	X	ERBR9: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
8	X	ERBR8: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
7	X	ERBR7: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
6	X	ERBR6: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
5	X	ERBR5: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
4	X	ERBR4: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
3	X	ERBR3: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
2	X	ERBR2: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
1	X	ERBR1: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
0	X	ERBR0: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

19.13.1.39 USB2_CONTROLLER_USB2D_ENDPTCOMPLETE_0

USB2D Endpoint Complete Register

Offset: 0x218 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ETCE15: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
30	0x0	ETCE14: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
29	0x0	ETCE13: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
28	0x0	ETCE12: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
27	0x0	ETCE11: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
26	0x0	ETCE10: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
25	0x0	ETCE9: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
24	0x0	ETCE8: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
23	0x0	ETCE7: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
22	0x0	ETCE6: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
21	0x0	ETCE5: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
20	0x0	ETCE4: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
19	0x0	ETCE3: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
18	0x0	ETCE2: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
17	0x0	ETCE1: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
16	0x0	ETCE0: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
15	0x0	ERCE15: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
14	0x0	ERCE14: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
13	0x0	ERCE13: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
12	0x0	ERCE12: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
11	0x0	ERCE11: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
10	0x0	ERCE10: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
9	0x0	ERCE9: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
8	0x0	ERCE8: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
7	0x0	ERCE7: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
6	0x0	ERCE6: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
5	0x0	ERCE5: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
4	0x0	ERCE4: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
3	0x0	ERCE3: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
2	0x0	ERCE2: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
1	0x0	ERCE1: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
0	0x0	ERCE0: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

19.13.1.40 USB2_CONTROLLER_USB2D_ENDPTCTRL0_0

USB2D Endpoint Control 0 Register

Offset: 0x21c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23	X	TXE: TX Endpoint Enable. Endpoint 0 is always enabled. 0 = DISABLE 1 = ENABLE
19:18	X	TXT: TX Endpoint Type. Endpoint 0 is fixed as a Control Endpoint. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
16	X	TXS: TX Endpoint Stall: Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0 = EP_OK 1 = EP_STALL
7	X	RXE: RX Endpoint Enable. Endpoint 0 is always enabled. 0 = DISABLE 1 = ENABLE
3:2	X	RXT: RX Endpoint Type. Endpoint 0 is fixed as a Control Endpoint. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
0	X	RXS: RX Endpoint Stall: Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0 = EP_OK 1 = EP_STALL

19.13.1.41 USB2_CONTROLLER_USB2D_ENDPTCTRLn_0

USB2D Endpoint Control 1 through 15 registers have the same field definitions and reset value. In the register offset, the value n is the register number (1 through 15).

USB2D Endpoint Control n Register

Offset: 0x220 + (n – 1) * 0x04 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL

Bit	R/W	Reset	Description
			1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

19.13.2 USB1 Controller Interface Registers

These are used to generate the actual RTL registers for USB1 controller interface.

19.13.2.1 USB1_IF_USB_SUSP_CTRL_0

This register controls the suspend and resume behavior of USB controller/PHY.

USB Suspend Control Register

Offset: 0x400 | Read/Write: R/W | Reset: 0bxxxxx00xxxxxx000xxx01000xx000000

Bit	R/W	Reset	Description
26	RW	0x0	FAST_WAKEUP_RESP: Enable Fast Response from UTMIP PHY for a Remote Wakeup request from device. This is used only for cases where wakeup response needs to be within 1 ms of spec. Used for Host mode ONLY. 0 = DISABLE 1 = ENABLE
25	RW	0x0	UTMIP_SUSPL1_SET: Enable SuspendL1 for UTMIP PHY. Enabling this will only cut off clocks to the UTMIP logic. The USB PLLs, PllU and UTMIP PLL will still be running.

Bit	R/W	Reset	Description
			0 = DISABLE 1 = ENABLE
18:16	RW	0x0	USB_WAKEUP_DEBOUNCE_COUNT: USB PHY wakeup debounce counter. USB will debounce any wakeup event by the number of clocks programmed in this counter. A value of 0 results in no debounce.
12	RW	0x0	UTMIP_PHY_ENB: Enable UTMIP PHY mode. Set this to 1 if using UTMIP PHY. Otherwise set this to 0. 0 = DISABLE 1 = ENABLE
11	RW	0x1	UTMIP_RESET: Reset going to UTMIP PHY (active high). This should be set to 1 whenever programming the UTMIP config registers. It should be cleared to 0 after the programming of UTMIP config registers is done. UTMIP config registers should be programmed only once before doing any transactions on USB. The UTMIP PHY registers should be programmed while UTMIP is in reset. 0 = DISABLE 1 = ENABLE
10	RW	0x0	USB_SUSP_POL: Polarity of the suspend signal going to USB PHY. 0 = Active low (default). 1 = Active high. This should not be changed by software. 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
9	RW	0x0	USB_PHY_CLK_VALID_INT_ENB: USB PHY clock valid interrupt enable. If this bit is enabled, interrupt is generated whenever USB clocks are resumed from a suspend. 0 = DISABLE 1 = ENABLE
8	RO	0x0	USB_PHY_CLK_VALID_INT_STS: USB PHY clock valid interrupt status. This bit is set whenever the USB PHY clock is woken up from suspend. Software must write a 1 to clear this bit. 0 = UNSET 1 = SET
7	RO	X	USB_PHY_CLK_VALID: USB PHY clock valid status. This bit indicates whether the USB PHY is generating a valid clock to the USB controller. If USB PHY clock is running, this bit is set to 1, else it is set to 0. 0 = UNSET 1 = SET
6	RO	X	USB_CLKEN: USB AHB clock enable status. Indicates whether the AHB clock to the USB controller is enabled or not. If AHB clock to USB controller is enabled, this bit is set to 1, else it is set to 0. NOTE: even when this is set to 0, all essential blocks that are required to resume USB clocks from suspend will be active and their AHB clock will not be suspended. 0 = UNSET 1 = SET
5	RW	0x0	USB_SUSP_CLR: Suspend Clear. Software must write a 1 to this bit to bring the PHY out of suspend mode. This is used when the software stops the PHY clock during suspend and then wants to initiate a resume. Software should also write 0 to clear it. NOTE: It is required that software generate a positive pulse on this bit to guarantee proper operation. 0 = UNSET 1 = SET
4	RW	0x0	USB_WAKE_ON_DISCON_EN_DEV: Wake on Disconnect Enable (device mode). When enabled (1), the USB device will wake up from suspend on a disconnect event. This is only valid when USB controller is in device mode, it is not applicable when USB controller is in host mode. 0 = DISABLE 1 = ENABLE
3	RW	0x0	USB_WAKE_ON_CNNT_EN_DEV: Wake on Connect Enable (device mode). When enabled (1), USB device will wake up from suspend on a connect event. This is only valid when USB controller is in device mode, it is not applicable when USB controller is in host mode. 0 = DISABLE 1 = ENABLE
2	RW	0x0	USB_WAKE_ON_RESUME_EN: Wake on resume enable. If this bit is enabled, USB will wake up from suspend whenever a resume event is detected on USB. This is valid for both USB device and USB host modes. 0 = DISABLE

Bit	R/W	Reset	Description
			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

19.13.2.2 USB1_IF_USB_PHY_VBUS_SENSORS_0

This register controls the OTG VBUS sensors in the USB PHY. There are 4 VBUS sensors:

- A_VBUS_VLD
- A_SESS_VLD
- B_SESS_VLD
- B_SESS_END

The debounced status of each sensor can be read from the corresponding _STS bit field of the sensor in this register. The _CHG_DET field is set to 1 whenever a change is detected in the value of the _STS bit field of the corresponding sensor. If _INT_EN is set, then an interrupt is generated to the processor. This interrupt can be routed to CPU/AVP by appropriately writing the USBD bits in the interrupt controller registers.

In case software wants to override the value for a sensor, it can set the corresponding _SW_EN to 1, and set the corresponding sensor _SW_VALUE to 1 or 0 as per the requirement.

There are two debouncers for each sensor - DEBOUNCE_A and DEBOUNCE_B. The debounce values for them are controlled by the register UTMIP_DEBOUNCE_CFG0, fields UTMIP_BIAS_DEBOUNCE_A and UTMIP_BIAS_DEBOUNCE_B. For each sensor, we can select whether to use DEBOUNCE_A or DEBOUNCE_B by setting the field _DEB_SEL_B to the appropriate value (SEL_A or SEL_B).

Note: Do not set either UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.

USB PHY VBUS SENSORS Control Register

Offset: 0x404 | Read/Write: R/W | Reset: 0bx0000x00x0000x00x0000x00x0000x00

Bit	R/W	Reset	Description
30	RW	0x0	A_VBUS_VLD_WAKEUP_EN: A_VBUS_VLD wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on A_VBUS_VLD. 0 = DISABLE 1 = ENABLE
29	RW	0x0	A_VBUS_VLD_DEB_SEL_B: A_VBUS_VLD debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
28	RW	0x0	A_VBUS_VLD_SW_VALUE: A_VBUS_VLD software value. Software should write the appropriate value (1/0) to set/unset the A_VBUS_VLD status. This is only valid when A_VBUS_VLD_SW_EN is

Bit	R/W	Reset	Description
			set. 0 = UNSET 1 = SET
27	RW	0x0	A_VBUS_VLD_SW_EN: A_VBUS_VLD software enable. Enable Software Controlled A_VBUS_VLD. Software sets this bit to drive the value in A_VBUS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
26	RO	X	A_VBUS_VLD_STS: A_VBUS_VLD status. This is set to 1 whenever A_VBUS_VLD sensor output is 1. 0 = UNSET 1 = SET
25	RO	0x0	A_VBUS_VLD_CHG_DET: A_VBUS_VLD change detect. This field is set by hardware whenever a change is detected in the value of A_VBUS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET
24	RW	0x0	A_VBUS_VLD_INT_EN: A_VBUS_VLD interrupt enable. If this field is set to 1, an interrupt is generated whenever A_VBUS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
22	RW	0x0	A_SESS_VLD_WAKEUP_EN: A_SESS_VLD wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on A_SESS_VLD. 0 = DISABLE 1 = ENABLE
21	RW	0x0	A_SESS_VLD_DEB_SEL_B: A_SESS_VLD debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
20	RW	0x0	A_SESS_VLD_SW_VALUE: A_SESS_VLD software value. Software should write the appropriate value (1/0) to set/unset the A_SESS_VLD status. This is only valid when A_SESS_VLD_SW_EN is set. 0 = UNSET 1 = SET
19	RW	0x0	A_SESS_VLD_SW_EN: A_SESS_VLD software enable. Enable Software Controlled A_SESS_VLD. Software sets this bit to drive the value in A_SESS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
18	RO	X	A_SESS_VLD_STS: A_SESS_VLD status. This is set to 1 whenever A_SESS_VLD sensor output is 1. 0 = UNSET 1 = SET
17	RO	0x0	A_SESS_VLD_CHG_DET: A_SESS_VLD change detect. This field is set by hardware whenever a change is detected in the value of A_SESS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET
16	RW	0x0	A_SESS_VLD_INT_EN: A_SESS_VLD interrupt enable. If this field is set to 1, an interrupt is generated whenever A_SESS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
14	RW	0x0	B_SESS_VLD_WAKEUP_EN: B_SESS_VLD wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on B_SESS_VLD. 0 = DISABLE 1 = ENABLE
13	RW	0x0	B_SESS_VLD_DEB_SEL_B: B_SESS_VLD debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A

Bit	R/W	Reset	Description
			1 = SEL_B
12	RW	0x0	B_SESS_VLD_SW_VALUE: B_SESS_VLD software value. Software should write the appropriate value (1/0) to set/unset the B_SESS_VLD status. This is only valid when B_SESS_VLD_SW_EN is set. 0 = UNSET 1 = SET
11	RW	0x0	B_SESS_VLD_SW_EN: B_SESS_VLD software enable. Enable Software Controlled B_SESS_VLD. Software sets this bit to drive the value in B_SESS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
10	RO	X	B_SESS_VLD_STS: B_SESS_VLD status. This is set to 1 whenever B_SESS_VLD sensor output is 1. 0 = UNSET 1 = SET
9	RO	0x0	B_SESS_VLD_CHG_DET: B_SESS_VLD change detect. This field is set by hardware whenever a change is detected in the value of B_SESS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET
8	RW	0x0	B_SESS_VLD_INT_EN: B_SESS_VLD interrupt enable. If this field is set to 1, an interrupt is generated whenever B_SESS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
6	RW	0x0	B_SESS_END_WAKEUP_EN: B_SESS_END wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on B_SESS_END. 0 = DISABLE 1 = ENABLE
5	RW	0x0	B_SESS_END_DEB_SEL_B: B_SESS_END debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
4	RW	0x0	B_SESS_END_SW_VALUE: B_SESS_END software value. Software should write the appropriate value (1/0) to set/unset the B_SESS_END status. This is only valid when B_SESS_END_SW_EN is set. 0 = UNSET 1 = SET
3	RW	0x0	B_SESS_END_SW_EN: B_SESS_END software enable. Enable Software Controlled B_SESS_END Software sets this bit to drive the value in B_SESS_END_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
2	RO	X	B_SESS_END_STS: B_SESS_END status. This is set to 1 whenever B_SESS_END sensor output is 1. 0 = UNSET 1 = SET
1	RO	0x0	B_SESS_END_CHG_DET: B_SESS_END change detect. This field is set by hardware whenever a change is detected in the value of B_SESS_END. software writes a 1 to clear it 0 = UNSET 1 = SET
0	RW	0x0	B_SESS_END_INT_EN: B_SESS_END interrupt enable. If this field is set to 1, an interrupt is generated whenever B_SESS_END_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE

19.13.2.3 USB1_IF_USB_PHY_VBUS_WAKEUP_ID_0

This register controls the battery charger (VDCD_DET, VDAT_DET), VBUS_WAKEUP and ID sensors. The following sensors are in this register:

- VBUS_WAKEUP
- ID
- VDAT_DET
- VDCD_DET

The debounced status of each sensor can be read from the corresponding _STS bit field of the sensor in this register. The _CHG_DET field is set to 1 whenever a change is detected in the value of the _STS bit field of the corresponding sensor. If _INT_EN is set, then an interrupt is generated to the processor. This interrupt can be routed to CPU/AVP by appropriately writing the USBD bits in the interrupt controller registers.

In case software wants to override the value for a sensor, it can set the corresponding _SW_EN to 1, and set the corresponding sensor _SW_VALUE to 1 or 0 as per the requirement.

There are two debouncers for each sensor - DEBOUNCE_A and DEBOUNCE_B. The debounce values for them are controlled by the register UTMIP_DEBOUNCE_CFG0, fields UTMIP_BIAS_DEBOUNCE_A and UTMIP_BIAS_DEBOUNCE_B. For each sensor, we can select whether to use DEBOUNCE_A or DEBOUNCE_B by setting the field _DEB_SEL_B to the appropriate value (SEL_A or SEL_B).

Note: Do not set either UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.

There are debouncers for VDAT_DET and VDCD_DET. These use separate debouncers -CHRG_DEBOUNCE_PERIOD_A and CHRG_DEBOUNCE_PERIOD_B. The debounce values for them are controlled by the register UTMIP_CHRG_DEB_CFG0, fields UTMIP_CHRG_DEBOUNCE_PERIOD_A and UTMIP_CHRG_DEBOUNCE_PERIOD_B. For each sensor, we can select whether to use CHRG_DEBOUNCE_PERIOD_A or CHRG_DEBOUNCE_PERIOD_B by setting the field _DEB_SEL_B to the appropriate value (SEL_A or SEL_B).

Note: Do not set either UTMIP_CHRG_DEBOUNCE_PERIOD_A or UTMIP_CHRG_DEBOUNCE_PERIOD_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.

USB PHY VBUS Wakeup and ID Control Register

Offset: 0x408 | Read/Write: R/W | Reset: 0b00000x00xx000x00xx000x00x1000x00

Bit	R/W	Reset	Description
31	RW	0x0	DIV_DET_EN: Battery charger divider detection enable. This goes to the USB2OTG pad. 0 = DISABLE 1 = ENABLE
30	RW	0x0	VBUS_WAKEUP_WAKEUP_EN: VBUS_WAKEUP wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on VBUS_WAKEUP. 0 = DISABLE 1 = ENABLE
29	RW	0x0	VDCD_DET_DEB_SEL_B: VCDT_DET debounce A/B select. Selects the debounce value from UTMIP_CHRG_DEBOUNCE_PERIOD_A or UTMIP_CHRG_DEBOUNCE_PERIOD_B from the register UTMIP_CHRG_DEB_CFG0. 0 = SEL_A 1 = SEL_B
28	RW	0x0	VDCD_DET_SW_VALUE: VDCD_DET software value. Software should write the appropriate value (1/0) to set/unset the VDCD_DET status. This is only valid when VDCD_DET_SW_EN is set. 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
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
23	RO	X	VOP_DIV2P7_DET: This read-only status bit is from the battery charging divider circuit of the USB2OTG pad 0 = UNSET 1 = SET
22	RO	X	VOP_DIV2P0_DET:This read-only status bit is from the battery charging divider circuit of the USB2OTG pad 0 = UNSET 1 = SET
21	RW	0x0	VDAT_DET_DEB_SEL_B: VDAT_DET debounce A/B select. Selects between the two debounce values UTMIP_CHRG_DEBOUNCE_PERIOD_A or UTMIP_CHRG_DEBOUNCE_PERIOD_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
20	RW	0x0	VDAT_DET_SW_VALUE: VDAT_DET software value. Software should write the appropriate value (1/0) to set/unset the VDAT_DET status. This is only valid when VDAT_DET_SW_EN is set. 0 = UNSET 1 = SET
19	RW	0x0	VDAT_DET_SW_EN: VDAT_DET software enable. Enable Software Controlled VDAT_DET. Software sets this bit to drive the value in VDAT_DET_SW_VALUE to the USB controller 0 = DISABLE 1 = ENABLE
18	RO	X	VDAT_DET_STS: VDAT_DET status. This is set to 1 whenever the VDAT_DET sensor output is 1. 0 = UNSET 1 = SET
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
15	RO	X	VON_DIV2P7_DET: This read-only status bit is from the battery charging divider circuit of the USB2OTG pad 0 = UNSET 1 = SET
14	RO	X	VON_DIV2P0_DET: This read-only status bit is from the battery charging divider circuit of the USB2OTG pad 0 = UNSET 1 = SET
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

19.13.2.4 USB1_IF_USB_PHY_ALT_VBUS_STS_0

USB PHY Alternate VBUS/ID Status Register

Offset: 0x40c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
14	X	ID_DIG_C_ALT: IDDIG_C alternate status 0 = UNSET 1 = SET
13	X	ID_DIG_C: IDDIG_C status 0 = UNSET 1 = SET
12	X	ID_DIG_B_ALT: IDDIG_B alternate status 0 = UNSET 1 = SET
11	X	ID_DIG_B: IDDIG_B status 0 = UNSET 1 = SET
10	X	ID_DIG_A_ALT: IDDIG_A alternate status 0 = UNSET 1 = SET
9	X	ID_DIG_A: IDDIG_A status 0 = UNSET 1 = SET
8	X	VDCD_DET_ALT: VDCD_DET alternate status 0 = UNSET 1 = SET
7	X	VDAT_DET_ALT: VDAT_DET alternate status 0 = UNSET 1 = SET
6	X	A_SESS_VLD_ALT: A_SESS_VLD alternate status 0 = UNSET 1 = SET
5	X	B_SESS_VLD_ALT: B_SESS_VLD alternate status 0 = UNSET 1 = SET
4	X	ID_DIG_ALT: ID alternate status 0 = UNSET 1 = SET
3	X	B_SESS_END_ALT: B_SESS_END alternate status 0 = UNSET 1 = SET
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

19.13.2.5 USB1_IF_USB_INTER_PKT_DELAY_CTRL_0

This controls the interpacket delay between two consecutive transmit packets in HS mode. This only applies to host mode as device never transmits two packets in a row.

Inter Packet Delay Control

Offset: 0x420 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx0010010

Bit	Reset	Description
6:0	0x12	IP_DELAY_TX2TX_HS: HS Tx to Tx inter-packet delay. Software should not change this.

19.13.2.6 USB1_IF_USB_RSM_DLY_0

USB Controller Resume Signaling Delay

Offset: 0x490 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0110100101111000

Bit	Reset	Description
15:0	0x6978	TIME_TO_RESUME: Send the resume back in number of 60 MHz cycles. Default gives 900 µs delay. Only applicable in host mode.

19.13.2.7 USB1_IF_SPARE_0

For ICUSB PADCTLS. Spare Register

Offset: 0x498 | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:16	0xffff	SPARE_HI: Spare register bits, defaulted to 1. SPARE_HI[0] - usb_port_reset_fix_en. Enable port-reset fix.
15:0	0x0	SPARE_LO: Spare register bits, defaulted to 0. SPARE_LO[4] - usb_hs_rsm_eop_fix_en, Enable HS_RESUME EOP fix. SPARE_LO[5] - usb_port_suspend_fix_en.

19.13.2.8 USB1_IF_USB1_NEW_CONTROL_0

USB Coherency and Memory Alignment Controls

Offset: 0x4c0 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx00001111xxxxxx00

Bit	Reset	Description
15:8	0xf	REQUEST_EXPIRY_COUNTER: Time to wait for coalescing the request
1	0x0	MEM_ALIGNMENT_MUX_EN: Mux to select between Tegra 3 style (0) and Tegra K1 style (1) DMA request generation mechanism 0 = DISABLE 1 = ENABLE
0	0x0	COHERENCY_EN: Enable fence mechanism. 0 = DISABLE 1 = ENABLE.

19.13.3 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

19.13.3.1 UTMIP REGISTER: PLL_CFG0

This register was used to configure PLL inside UTMIP block prior to Tegra K1 devices. This has been de-featured from UTMIP space and moved to CAR space. This register is used to configure the PHY PLL contained in the UTMIP module. Refer to the Clock and Reset Controller section for details on this register.

> **Note:** This register has been moved to the Clock and Reset section of this document. PROGRAMMING THIS REGISTER HERE WILL HAVE NO EFFECT!

19.13.3.2 USB1_UTMIP_PLL_CFG1_0

UTMIP PLL and PLLU Configuration Register 1

> **Note:** This register has been moved to the Clock and Reset section of this document. PROGRAMMING THIS REGISTER HERE WILL HAVE NO EFFECT!

This register was used to configure the PLL inside the UTMIP block prior to Tegra K1 devices. This register has been defeatured from UTMIP space and moved to CAR space. Refer to the Clock and Reset Controller section for details on this register.

19.13.3.3 USB1_UTMIP_XCVR_CFG0_0

UTMIP Transceiver Cell Configuration Register 0

Offset: 0x808 | Read/Write: R/W | Reset: 0b00100000001001010110010100000000

Bit	Reset	Description
31:25	0x10	UTMIP_XCVR_HSSLEW_MSB: Most significant bits of HS_SLEW.
24:22	0x0	UTMIP_XCVR_SETUP_MSB: Most significant bits of SETUP.
21	0x1	UTMIP_XCVR_LSBIAS_SEL: Low speed bias selection method for USB transceiver pad
20	0x0	UTMIP_XCVR_DISCON_METHOD: Disconnect method on the USB transceiver pad
19	0x0	UTMIP_FORCE_PDZI_POWERUP: Force PDZI input into power up.
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.

19.13.3.4 USB1_UTMIP_BIAS_CFG0_0

UTMIP Bias Cell Configuration Register 0

Offset: 0x80c | Read/Write: R/W | Reset: 0bx00000001100000000000110000000000

Bit	Reset	Description
30	0x0	UTMIP_IDDIG_C_VAL: See IDDIG_C_SEL.
29	0x0	UTMIP_IDDIG_C_SEL: 0: IdDig_c = IdDig_c. 1: IdDig_c = IDDIG_C_VAL.
28	0x0	UTMIP_IDDIG_B_VAL: See IDDIG_B_SEL.
27	0x0	UTMIP_IDDIG_B_SEL: 0: IdDig_b = IdDig_b. 1: IdDig_b = IDDIG_B_VAL.
26	0x0	UTMIP_IDDIG_A_VAL: See IDDIG_A_SEL.
25	0x0	UTMIP_IDDIG_A_SEL: 0: IdDig_a = IdDig_a. 1: IdDig_a = IDDIG_A_VAL.
24	0x0	UTMIP_HSDISCON_LEVEL_MSB: Most significant bit of UTMIP_HSDISCON_LEVEL, bit 2
23	0x1	UTMIP_IDPD_VAL: See IDPD_SEL.
22	0x1	UTMIP_IDPD_SEL: 0: Reserved. Refer to the PMC registers for this feature.
21	0x0	UTMIP_IDDIG_VAL: See IDDIG_SEL.
20	0x0	UTMIP_IDDIG_SEL: 0: IdDig = IdDig. 1: IdDig = IDDIG_VAL.
19	0x0	UTMIP_GPI_VAL: See GPI_SEL.
18	0x0	UTMIP_GPI_SEL: 0: StaticGpi = IdDig. 1: StaticGpi = GPI_VAL.
17:15	0x0	UTMIP_ACTIVE_TERM_OFFSET: Active termination control offset.
14:12	0x0	UTMIP_ACTIVE_PULLUP_OFFSET: Active 1.5K pullup control offset.
11	0x1	UTMIP_OTGPD: Power down OTG circuit.
10	0x1	UTMIP_BIASPD: Power down bias circuit.
9:8	0x0	UTMIP_VBUS_LEVEL_LEVEL: Vbus detector level.
7:6	0x0	UTMIP_SESS_LEVEL_LEVEL: SessionEnd detector level.
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.

19.13.3.5 USB1_UTMIP_HSRX_CFG0_0

UTMIP High Speed Receive Config 0

Offset: 0x810 | Read/Write: R/W | Reset: 0b10010001011001010011010000000000

Bit	Reset	Description
31:30	0x2	UTMIP_KEEP_PATT_ON_ACTIVE: Keep the stay alive pattern on active
29	0x0	UTMIP_ALLOW_CONSEC_UPDN: Allow consecutive ups and downs on the bits, debug only, set to 0.
28	0x1	UTMIP_REALIGN_ON_NEW_PKT: Realign the inertia counters on a new packet
27:24	0x1	UTMIP_PCOUNT_UPDN_DIV: The number of (edges-1) needed to move the sampling point
23:21	0x3	UTMIP_SQUELCH_EOP_DLY: Limit the delay of the squelch at EOP time
20	0x0	UTMIP_NO_STRIPPING: Do not strip incoming data
19:15	0xa	UTMIP_IDLE_WAIT: Number of idle cycles to declare IDLE.

Bit	Reset	Description
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

19.13.3.6 USB1_UTMIP_HSRX_CFG1_0

UTMIP High Speed Receive Config 1

Offset: 0x814 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxx010011

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

19.13.3.7 USB1_UTMIP_FSLSRX_CFG0_0

UTMIP Full and Low Speed Receive Config 0

Offset: 0x818 | Read/Write: R/W | Reset: 0b11111101010101001000010000101001

Bit	Reset	Description
31	0x1	UTMIP_FSLS_SE1_DRIBBLE_FILTER: One SE1, do not allow dribble
30	0x1	UTMIP_FSLS_SE1_FILTER: Filter SE1
29	0x1	UTMIP_FSLS_SERIAL_SE0_RCV
28:26	0x7	UTMIP_FSLS_UPR_DRIBBLE_SIZE: Do not allow <= dribble bits
25:23	0x2	UTMIP_FSLS_LWR_DRIBBLE_SIZE: Do not allow >= dribble bits
22	0x1	UTMIP_FSLS_EOP_ENDS_AT_SE0: Only look for transitioning out of EOP
21:16	0x14	UTMIP_FSLS_KCOUNT_MAX: Number of K bits in question
15	0x1	UTMIP_FSLS_KCOUNT_LIMIT: Limit the number of bit times a K can last
14	0x0	UTMIP_FSLS_ACTIVE_ON_FULL_SYNC: Require a full sync pattern to declare the data received
13:8	0x4	UTMIP_FSLS_IDLE_WAIT_MAX: 4 bits of SEO should exceed the time limit
7	0x0	UTMIP_FSLS_IDLE_WAIT_LIMIT: Enable the reset of the state machine on extended SE0
6:1	0x14	UTMIP_FSLS_IDLE_COUNT_MAX: 20 bits of idle should end the packet if FsLsIdleCountLimitCfg=1.
0	0x1	UTMIP_FSLS_IDLE_COUNT_LIMIT: Give up on packet if a long sequence of J

19.13.3.8 USB1_UTMIP_FSLSRX_CFG1_0

UTMIP Full and Low Speed Receive Config 1

Offset: 0x81c | Read/Write: R/W | Reset: 0bxxxxx01000100110011101000000000

Bit	Reset	Description
26	0x0	UTMIP_EARLY_LINE_STATE_FILTER: Assumes line state filtering table is inclusive, not exclusive
25:23	0x4	UTMIP_LS_BOUNCE_LENGTH: Number of clock cycle of LS stable
22:17	0x13	UTMIP_LS_EXTRACTION_COUNT: Phase count on which LS bits are extracted
16:11	0xe	UTMIP_LS_EOP_START_COUNT: Number of SEO clock cycles to block bit extraction
10:5	0x20	UTMIP_LS_SE0_COUNT: Only for this number of 60MHz of SEO and Idle to end packet
4	0x0	UTMIP_LS_LENIENT_DRIBBLE: Allow for large dribble in low speed mode
3	0x0	UTMIP_FS_LENIENT_DRIBBLE: Allow for large dribble in full speed mode
2	0x0	UTMIP_FS_WEAK_SYNC: Only look for a KK pattern, instead of KJKK
1	0x0	UTMIP_FS_DEBOUNCE: Whether full speed uses debouncing
0	0x0	UTMIP_FS_EOP_LENGTH: Whether full speed EOP is determined within 3(0) or 4(1) 60MHz cycles

19.13.3.9 USB1_UTMIP_TX_CFG0_0

UTMIP Transmit Config Signals

Offset: 0x820 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxx00010000001000000000

Bit	Reset	Description
19	0x0	UTMIP_FS_PREAMBLE_J: Output enable sends an initial J before sync pattern
18	0x0	UTMIP_FS_POSTAMBLE_OUTPUT_ENABLE: Output enable turns off 1/2 cycle after
17	0x0	UTMIP_FS_PREAMBLE_OUTPUT_ENABLE: Output enable turns on 1/2 cycle before
16	0x1	UTMIP_FSLS_ALLOW_SOP_TX_STUFF_ERR: Allow SOP to be source of transmit error stuffing
15	0x0	UTMIP_HS_READY_WAIT_FOR_VALID
14:10	0x0	UTMIP_HS_TX_IPG_DLY
9	0x1	UTMIP_HS_DISCON_EOP_ONLY: Only check during EOP
8	0x0	UTMIP_HS_DISCON_DISABLE: Disable high speed disconnect
7	0x0	UTMIP_HS_POSTAMBLE_OUTPUT_ENABLE: Output enable turns off 1 cycle after
6	0x0	UTMIP_HS_PREAMBLE_OUTPUT_ENABLE: Output enable turns on 1 cycle before
5	0x0	UTMIP_SIE_RESUME_ON_LINESTATE: SIE, not macrocell, detects LineState change to resume
4	0x0	UTMIP_SOF_ON_NO_STUFF: SOF when OpMode 3 -- perhaps, when sending controller made packets
3	0x0	UTMIP_SOF_ON_NO_ENCODE: SOF when OpMode 2 -- not likely, for Chirp
2	0x0	UTMIP_NO_STUFFING: No bit stuffing, static programming
1	0x0	UTMIP_NO_ENCODING: No encoding, static programming
0	0x0	UTMIP_NO_SYNC_NO_EOP: Do not send SYNC or EOP

19.13.3.10 USB1_UTMIP_MISC_CFG0_0

UTMIP Miscellaneous Configurations

Offset: 0x824 | Read/Write: R/W | Reset: 0bx00000111110000000000000001111000

Bit	Reset	Description
30:27	0x0	UTMIP_DPDM_OBSERVE_SEL: Select DP/DM obs signals
26	0x0	UTMIP_DPDM_OBSERVE: Use DP/DM as obs bus
25	0x1	UTMIP_KEEP_XCVR_PD_ON_SOFT_DISCON
24	0x1	UTMIP_ALLOW_LS_ON_SOFT_DISCON
23	0x1	UTMIP_FORCE_FS_DISABLE_ON_DEV_CHIRP
22	0x1	UTMIP_SUSPEND_EXIT_ON_EDGE: Suspend exit requires edge or simply a value.
21	0x1	UTMIP_LS_TO_FS_SKIP_4MS: Do not block changes for 4 ms when going from LS to FS (should not happen)
20:19	0x0	UTMIP_INJECT_ERROR_TYPE: Force error insertion into RX path. (Used for IOBIST.) 0 = DISABLE 1 = BIT_ERR 2 = RX_ERR 3 = BIT_RX_ERR
18	0x0	UTMIP_FORCE_HS_CLOCK_ON: Force HS clock always on.
17	0x0	UTMIP_DISABLE_HS_TERM: Force HS termination inactive.
16	0x0	UTMIP_FORCE_HS_TERM: Force HS termination active.
15	0x0	UTMIP_DISABLE_PULLUP_DP: Force DP pullup inactive. (Overrides FORCE_PULLUP_DP.)
14	0x0	UTMIP_DISABLE_PULLUP_DM: Force DM pullup inactive. (Overrides FORCE_PULLUP_DM.)
13	0x0	UTMIP_DISABLE_PULLDN_DP: Force DP pulldown inactive. (Overrides FORCE_PULLDN_DP.)
12	0x0	UTMIP_DISABLE_PULLDN_DM: Force DM pulldown inactive. (Overrides FORCE_PULLDN_DM.)
11	0x0	UTMIP_FORCE_PULLUP_DP: Force DP pullup active.
10	0x0	UTMIP_FORCE_PULLUP_DM: Force DM pullup active.
9	0x0	UTMIP_FORCE_PULLDN_DP: Force DP pulldown active.
8	0x0	UTMIP_FORCE_PULLDN_DM: Force DM pulldown active.
7:5	0x3	UTMIP_STABLE_COUNT: Number of cycles of crystal clock of signal not changing to consider stable.
4	0x1	UTMIP_STABLE_ALL: Determines if all signal need to be stable to not change a config.
3	0x1	UTMIP_NO_FREE_ON_SUSPEND: Do not use free running terminations during suspend.
2	0x0	UTMIP_NEVER_FREE_RUNNING_TERMS: Ignore free-running terminations, even when no clock
1	0x0	UTMIP_ALWAYS_FREE_RUNNING_TERMS: Use free-running terminations at all time
0	0x0	UTMIP_COMB_TERMS: Use combinational terminations or synced through CLKXTAL

19.13.3.11 USB1_UTMIP_MISC_CFG1_0

UTMIP Miscellaneous Configurations

Offset: 0x828 | Read/Write: R/W | Reset: 0bx10000000001100110000000000100100

Bit	Reset	Description
30	0x1	UTMIP_PHY_XTAL_CLOCKEN: Selects whether to enable the crystal clock in the module.

Bit	Reset	Description
29	0x0	UTMIP_LINESTATE_BYPASS: Bypass LineState reclocking logic
28	0x0	UTMIP_LINESTATE_NEG: Use negative edge sync for line state
27	0x0	UTMIP_LINESTATE_XCVRSEL3: 0: Use FS filtering on line state when XcvrSel=3 1: Use LS filtering on line state when XcvrSel=3
26:25	0x0	UTMIP_OBS_SEL
24	0x0	UTMIP_FSLS_TDM
23	0x0	UTMIP_FORCE_IOBIST_CLK_ON
22:18	0x6	UTMIP_PLL_ACTIVE_DLY_COUNT: Reserved. Config moved to the Clock and Reset space.
17:6	0x600	UTMIP_PLLU_STABLE_COUNT: Reserved. Moved to the Clock and Reset space.
5	0x1	UTMIP_RX_ERROR_CNT_CLR
4	0x0	UTMIP_RX_ERROR_CNT_EN
3	0x0	UTMIP_FLIP_FSLS_POLARITY
2	0x1	UTMIP_SUSPEND_TERMSEL
1:0	0x0	UTMIP_XCVRSEL3: Bit 0: 0: 0xa5 -> treat as KeepAlive 1: treat as regular packet Bit 1: 0: Turn on FS EOP detection 1: Turn off FS EOP detection

19.13.3.12 USB1_UTMIP_DEBOUNCE_CFG0_0

UTMIP Avalid and Bvalid Debounce

Debounce values IdDig, Avalid, Bvalid, VbusValid, VbusWakeUp, and SessEnd. Each of these signals has its own debouncer, and for each of those one out of 2 debouncing times can be chosen (BIAS_DEBOUNCE_A or BIAS_DEBOUNCE_B.)

The values of DEBOUNCE_A and DEBOUNCE_B are calculated as follows:

0xffff -> No debouncing at all

ms = *1000 / (1/19.2MHz) / 4

So to program a 1 ms debounce for BIAS_DEBOUNCE_A:

BIAS_DEBOUNCE_A[15:0] = 1000 * 19.2 / 4 = 4800 = 0x12c0

Offset: 0x82c | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:16	0xffff	UTMIP_BIAS_DEBOUNCE_B: simulation value -- Used for interrupts
15:0	0xffff	UTMIP_BIAS_DEBOUNCE_A: simulation value -- Used for interrupts

19.13.3.13 USB1_UTMIP_BAT_CHRG_CFG0_0

UTMIP Battery Charger Configuration

Offset: 0x830 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxx000000xx000001

Bit	Reset	Description
13:8	0x0	UTMIP_CHRG_DEBOUNCE_TIMESCALE: Debouncer time scaling, factor-1 to slow down debouncing by. So 0 is 1, 1 is 2, etc.
5	0x0	UTMIP_OP_I_SRC_EN
4	0x0	UTMIP_ON_SRC_EN

Bit	Reset	Description
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

19.13.3.14 USB1_UTMIP_SPARE_CFG0_0

UTMIP Spare Configuration Bits

Offset: 0x834 | Read/Write: R/W | Reset: 0b11111111111111110000000011111000

Bit	Reset	Description
31:0	-65288	UTMIP_SPARE: Spare register bits 0: HS_RX_IPG_ERROR_ENABLE 1: HS_RX_FLUSH_ALAP. Flush as late as possible 2: HS_RX_LATE_SQUELCH. Delay Squelch by 1 CLK480 cycle. 3: FUSE_SETUP_SEL. Select between regular CFG value and JTAG values for UX_SETUP 4: FUSE_TERM_RANGE_ADJ_SEL. Select between regular CFG value and JTAG values for UX_TERM_RANGE_ADJ 5: FUSE_SPARE. Select between regular CFG value and JTAG values. For any ECOs. 6: FUSE_HS_SQUELCH_LEVEL. Select between regular CFG value and JTAG values 7: FUSE_HS_IREF_CAP_CFG. Select between regular CFG value and JTAG values 31 to 8: Reserved

19.13.3.15 USB1_UTMIP_XCVR_CFG1_0

UTMIP Transceiver Cell Configuration Register 1

Offset: 0x838 | Read/Write: R/W | Reset: 0bxxxx00000001100010000001000010101

Bit	Reset	Description
27:26	0x0	UTMIP_XCVR_RPU_RANGE_ADJ: 1.5k pull-up resistor range shift.
25:24	0x0	UTMIP_XCVR_HS_IREF_CAP: High-speed Iref cap control for bias current stability.
23:22	0x0	UTMIP_XCVR_SPARE: Spare bits for USB transceiver pad.
21:18	0x6	UTMIP_XCVR_TERM_RANGE_ADJ: Range adjustment on terminations.
17	0x0	UTMIP_RCTRL_SW_SET: Use a software override on RCTRL instead of automatic bias control.
16:12	0x8	UTMIP_RCTRL_SW_VAL: Encoded value to use on RCTRL when software override is enabled, 0 to 16 only.
11	0x0	UTMIP_TCTRL_SW_SET: Use a software override on TCTRL instead of automatic bias control
10:6	0x8	UTMIP_TCTRL_SW_VAL: Encoded value to use on TCTRL when software override is enabled, 0 to 16 only.
5	0x0	UTMIP_FORCE_PDDR_POWERUP: Force PDDR input into power up.
4	0x1	UTMIP_FORCE_PDDR_POWERDOWN: Force PDDR input into power down. (Overrides FORCE_PDDR_POWERUP.)
3	0x0	UTMIP_FORCE_PDCHRP_POWERUP: Force PDCHRP input into power up.
2	0x1	UTMIP_FORCE_PDCHRP_POWERDOWN: Force PDCHRP input into power down. (Overrides FORCE_PDCHRP_POWERUP.)
1	0x0	UTMIP_FORCE_PDDISC_POWERUP: Force PDDISC input into power up.

Bit	Reset	Description
0	0x1	UTMIP_FORCE_PDDISC_POWERDOWN: Force PDDISC input into power down. (Overrides FORCE_PDDISC_POWERUP.)

19.13.3.16 USB1_UTMIP_BIAS_CFG1_0

UTMIP Bias Cell Configuration Register 1

Offset: 0x83c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxx00000000101101

Bit	Reset	Description
13:8	0x0	UTMIP_BIAS_DEBOUNCE_TIMESCALE: Debouncer time scaling. Slows down debouncing by a factor-1. So 0 is 1, 1 is 2, etc.
7:3	0x5	UTMIP_BIAS_PDTRK_COUNT: Control the BIAS cell power down lag. The lag should be 20 µs. For a crystal clock of 13 MHz, it should be set to 5.
2	0x1	UTMIP_VBUS_WAKEUP_POWERDOWN: Reserved. See the PMC registers for this functionality.
1	0x0	UTMIP_FORCE_PDTRK_POWERUP: Force PDTRK input into power up.
0	0x1	UTMIP_FORCE_PDTRK_POWERDOWN: Force PDTRK input into power down. (Overrides FORCE_PDTRK_POWERUP.)

19.13.3.17 USB1_UTMIP_BIAS_STS0_0

UTMIP Bias Cell Status Register 0

Offset: 0x840 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	UTMIP_TCTRL: Thermal encoding output from USB bias pad.
15:0	X	UTMIP_RCTRL: Thermal encoding output from USB bias pad.

19.13.3.18 USB1_UTMIP_CHRG_DEB_CFG0_0

UTMIP VDcd_Det and VDat_Det Debounce

Debounce values VDcd_Det and VDat_Det. Each of these signals has its own debouncer and for each of those, 1 out of 2 debouncing times can be chosen (CHRG_DEBOUNCE_PERIOD_A or CHRG_DEBOUNCE_PERIOD_B).

The values of DEBOUNCE_PERIOD_A and DEBOUNCE_PERIOD_B are calculated as follows:

0xffff -> No debouncing at all

ms = *1000 / (1/19.2MHz) / 4

So to program a 1 ms debounce for CHG_DEBOUNCE_PERIOD:

CHG_DEBOUNCE_PERIOD[15:0] = 1000 * 19.2 / 4 = 4800 = 0x12c0

Offset: 0x844 | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:16	0xffff	UTMIP_CHRG_DEBOUNCE_PERIOD_B: Simulation value -- Used for interrupts
15:0	0xffff	UTMIP_CHRG_DEBOUNCE_PERIOD_A: Simulation value -- Used for interrupts

19.13.3.19 USB1_UTMIP_MISC_STS0_0

UTMIP MISC Status Register

Offset: 0x848 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
21	X	UTMIP_FS_DRV_EN: Indicates when the controller is driving on the bus
20:0	X	UTMIP_SPARE_FUSES: Spare Fuses value to keep the connections preserved

19.13.3.20 USB1_UTMIP_PMC_WAKEUP0_0

UTMIP PMC Wakeup value

Offset: 0x84c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	R/W	Reset	Description
1	RO	X	UTMIP_LINE_WAKEUP_EVENT_INT_STS: Indicates the status of the PMC wakeup event
0	RW	0x0	UTMIP_LINE_WAKEUP_EVENT_INT_ENB: Interrupt mask for the PMC wakeup event

19.13.3.21 USB2_QH_USB2D_QH_EP_n_OUT_0

USB2D Queue Head for OUT Endpoint n

There are 16 USB2D Queue Head for OUT Endpoint registers, where n = 0 through 15.

Offset: 0x1000 + (n * 0x80) | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint n. 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 n.

19.13.3.22 USB2_QH_USB2D_QH_EP_n_IN_0

USB2D Queue Head for IN Endpoint n

There are 16 USB2D Queue Head for IN Endpoint registers, where n = 0 through 15.

Offset: 0x1040 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint n. 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 n.

19.13.4 USB2 Controller Registers

19.13.4.1 USB2_CONTROLLER_1_USB2D_ID_0

USB2D Identification Register

Offset: 0x0 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	CIVERSION: Identifies the CI version

Bit	Reset	Description
28:25	X	VERSION: Identifies the version of the core
24:21	X	REVISION: Revision number of the USB controller. This is set to 0x0.
20:16	X	TAG: Identifies the tag of the core
15:8	X	NID: One's complement version of ID. This field is set to 0xF9.
7:0	X	ID: Configuration number. This field is set to 0x06

19.13.4.2 USB2_CONTROLLER_1_USB2D_HW_HOST_0

USB2D Hardware Host Register

Offset: 0x8 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
3:1	X	NPORT: VUSB_HS_NUM_PORT-1: This host controller has only 1 port. So this field will always be 0.
0	X	HC: VUSB_HS_HOST: Indicates support for host mode. Set to 1.

19.13.4.3 USB2_CONTROLLER_1_USB2D_HW_DEVICE_0

USB2D Hardware Device Register

Offset: 0xc | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
5:1	X	DEVEP: VUSB_HS_DV_EP: Number of endpoints supported by this device controller. Set to 16. This includes control endpoint 0.
0	X	DC: Device capable: Set to 1 indicating support for device mode.

19.13.4.4 USB2_CONTROLLER_1_USB2D_HW_TXBUF_0

USB2D Hardware TX Buffer Register

Offset: 0x10 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	TXCHANADD: VUSB_HS_TX_CHAN_ADD: Total number of address bits for the transmit buffer of each transmit endpoint. Set to 7. Each transmit buffer is 128 words deep.
15:8	X	TXADD: VUSB_HS_TX_ADD: Total number of address bits for the transmit buffer. Set to 11. The total depth of the transmit buffer is 2048 words.
7:0	X	TCBURST: VUSB_HS_TX_BURST: Maximum burst size supported by the transmit endpoints for data transfers. Set to 8.

19.13.4.5 USB2_CONTROLLER_1_USB2D_HW_RXBUF_0

USB2D RX Buffer HW Parameters Register

Offset: 0x14 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:8	X	RXADD: VUSB_HS_RX_ADD: Total number of address bits for the receive buffer. Set to 7. The total depth of the receive buffer is 128 words.

Bit	Reset	Description
7:0	X	RXBURST: VUSB_HS_RX_BURST: Maximum burst size supported by the receive endpoints for data transfers. Set to 8.

19.13.4.6 USB2_CONTROLLER_1_USB2D_GPTIMER0LD_0

The host/device controller drivers can measure time-related activities using these timer registers. These registers are not part of the standard EHCI controller.

Offset: 0x80 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000)

Bit	Reset	Description
23:0	0x0	GPTIMER0LD: This field has the value to be loaded into the GPTCNT countdown timer on a reset action. The value in this register represents the time in microseconds minus 1 for the timer duration .

19.13.4.7 USB2_CONTROLLER_1_USB2D_GPTIMER0CTRL_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00XXXXXX (0b00xxxxx0xxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	0x0	GTPRUN: This bit enables the general-purpose timer to run. Setting or clearing this bit will not have an effect on the GPTCNT counter value.
30	WO	0x0	GPTRST: Writing a one to this bit reloads the GPTCNT with the value in GPTLD.
24	RW	0x0	GPTMODE: This bit selects between a single timer countdown and a looped countdown. In one-shot mode, the timer counts down to zero, generates an interrupt, and stops until the counter is reset by software. In repeat mode, the timer counts down to zero, generates an interrupt, and automatically reloads the counter to begin again.
23:0	RO	X	GPTCNT: This field has the value of the running timer.

19.13.4.8 USB2_CONTROLLER_1_USB2D_GPTIMER1LD_0

Offset: 0x88 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000)

Bit	Reset	Description
23:0	0x0	GPTIMER1LD: This field has the value to be loaded into the GPTCNT countdown timer on a reset action. The value in this register represents the time in microseconds minus 1 for the timer duration .

19.13.4.9 USB2_CONTROLLER_1_USB2D_GPTIMER1CTRL_0

Offset: 0x8c | Read/Write: R/W | Reset: 0x00XXXXXX (0b00xxxxx0xxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	0x0	GTPRUN: This bit enables the general-purpose timer to run. Setting or clearing this bit will not have an effect on the GPTCNT counter value.
30	WO	0x0	GPTRST: Writing a one to this bit reloads the GPTCNT with the value in GPTLD.
24	RW	0x0	GPTMODE: This bit selects between a single timer countdown and a looped countdown. In one-shot mode, the timer counts down to zero, generates an interrupt, and stops until the counter is reset by software. In repeat mode, the timer counts down to zero, generates an interrupt, and automatically reloads the counter to begin again.
23:0	RO	X	GPTCNT: This field has the value of the running timer.

19.13.4.10 USB2_CONTROLLER_1_USB2D_CAPLENGTH_0

USB2D Capability Register Length Register

Offset: 0x100 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
7:0	X	CAPLENGTH: Indicates which offset to add to the register base address at the beginning of the Operational Register. Set to 0x30.

19.13.4.11 USB2_CONTROLLER_1_USB2D_HCIVERSON_0

USB2D Host Interface Version Number Register

Offset: 0x102 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	HCIVERSION: Contains a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision. This host controller supports EHCI revision 1.00.

19.13.4.12 USB2_CONTROLLER_1_USB2D_HCSPARAMS_0

USB2D Host Control Structural Parameters Register

Offset: 0x104 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
27:24	X	N_TT: Number of Transaction Translators: Indicates the number of embedded transaction translators associated with the USB2.0 host controller. This field is always set to 1 indicating only 1 embedded TT is implemented in this implementation. This is a non-EHCI field to support embedded TT.
23:20	X	N_PTT: Number of Ports per Transaction Translator: Indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. Field always equals N_PORTS. This is a non-EHCI field to support embedded TT.
15:12	X	N_CC: Number of Companion Controller: Indicates the number of companion controllers. This field is set to 0.
11:8	X	N_PCC: Number of Ports per Companion Controller: Indicates the number of ports supported per internal companion controller. This field is set to 0.
4	X	PPC: Port Power Control: Indicates whether the host controller implementation includes port power control. 1 = Ports have port power switches 0= Ports do not have port power switches. This field affects the functionality of the port Power field in each port status and control register. This field is set to 1.
3:0	X	N_PORTS: Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller. This field is fixed to 1, since this host controller only supports 1 port.

19.13.4.13 USB2_CONTROLLER_1_USB2D_HCCPARAMS_0

USB2D Host Control Capability Parameters Register

Offset: 0x108 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
18	X	PPC: Per-Port Change Event Capability. Default = 1b. This field indicates the support for per-port change events. This field is related to the USBCMD PPE field, USBSTS PPCI field, and USBINTR PPCE field.

Bit	Reset	Description
17	X	LEN: Link Power Management Capability. Default = 1b. This field indicates the support for LPM L1 state. This field is related to the USBCMD HIRD field, POSTSCx SSTS and DA fields, and HOSTPCx LEN, BA, and EPLPM fields.
15:8	X	EECP: EHCI Extended Capabilities Pointer: Indicates a capabilities list exists. A value of 00h indicates no extended capabilities are implemented. For this implementation this field is always "0".
7:4	X	IST: Isochronous Scheduling Threshold. This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit [7] is zero, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit [7] is a one, then host software assumes the host controller may cache an isochronous data structure for an entire frame. This field will always be "0".
2	X	ASP: Asynchronous Schedule Park Capability. 1 = (Default) the host controller supports the park feature for high-speed queue heads in the Asynchronous Schedule. The feature can be disabled or enabled and set to a specific level by using the Asynchronous Schedule Park Mode Enable and Asynchronous Schedule Park Mode Count fields in the USBCMD register. This field is always 1.
1	X	PFL: Programmable Frame List Flag. 0 = System software must use a frame list length of 1024 elements with this host controller. The USBCMD register Frame List Size field is a read-only register and must be set to zero. 1 = System software can specify and use a smaller frame list and configure the host controller via the USBCMD register Frame List Size field. The frame list must always be aligned on a 4K-page boundary. This requirement ensures that the frame list is always physically contiguous. This field will always be "1".

19.13.4.14 USB2_CONTROLLER_1_USB2D_DCIVERSION_0

USB2D Device Interface Version Number Register

Offset: 0x120 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	DCIVERSION: The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.

19.13.4.15 USB2_CONTROLLER_1_USB2D_DCCPARAMS_0

USB2D Device Control Capabilities Register

Offset: 0x124 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
8	X	HC: Host Capable: 1 = This controller is capable of operating as an EHCI compatible USB 2 0 host controller operating as an EHCI compatible USB 2.0 host controller. This field is set to 1.
7	X	DC: Device Capable: 1 = Controller is capable of operating as USB 2.0 device. This field is set to 1
5	X	LEN: Link Power Management Capability - RO. Default = 1b. This field indicates the support for LPM L1 state. This field is related to the DEVLCx ASUS, STL,BA and NYT fields.
4:0	X	DEN: Device Endpoint Number: Number of endpoints built into the device controller. This is set to 16.

19.13.4.16 USB2_CONTROLLER_1_USB2D_EXTSTS_0

USB2D EXTSTS Register

Offset: 0x128 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000x

Bit	R/W	Reset	Description
4	RW	0x0	TI1: General Purpose Timer Interrupt 1 - RWC. Default = 0b. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero. Writing a one to this bit will clear it.
3	RW	0x0	TI0: General Purpose Timer Interrupt 0 - RWC. Default = 0b. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero. Writing a one to this bit will clear it.
2	RW	0x0	UPA: USB Host Periodic Interrupt (USBHSTPERINT) R/WC. This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set and the TD was from the periodic schedule. This bit is also set by the Host Controller when a short packet is detected AND the packet is on the periodic schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be zero. 0 = DISABLE 1 = ENABLE
1	RW	0x0	UAI: USB Host Asynchronous Interrupt (USBHSTASYNCINT) R/WC. This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set AND the TD was from the asynchronous schedule. This bit is also set by the Host when a short packet is detected AND the packet is on the asynchronous schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be zero. 0 = DISABLE 1 = ENABLE
0	RO	X	NAKI: NAK Interrupt Bit Read Only. This bit is read only. It is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and the corresponding TX/RX Endpoint NAK Enable bit are set. This bit is automatically cleared by hardware when the all the enabled TX/RX Endpoint NAK bits are cleared. 0 = DISABLE 1 = ENABLE

19.13.4.17 USB2_CONTROLLER_1_USB2D_USBEXTINTR_0

USB2D EXTINTR Register

Offset: 0x12c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000

Bit	Reset	Description
4	0x0	TIE1: General Purpose Timer Interrupt Enable 1 - RWC. Default = 0b. When this bit is a one, and the TI1 bit in the EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the TI1 bit.
3	0x0	TIE0: General Purpose Timer Interrupt Enable 0 - RWC. Default = 0b. When this bit is a one, and the TI1 bit in the EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the TI1 bit.
2	0x0	UPIE: UPIE Interrupt Enable. 1 = USB controller issues an interrupt if the UPA bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE
1	0x0	UAIE: UAIE Interrupt Enable. 1 = USB controller issues an interrupt if the UAI bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE
0	0x0	NAKE: NAK Interrupt Enable. 1 = USB controller issues an interrupt if the NAKI bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE

19.13.4.18 USB2_CONTROLLER_1_USB2D_USBCMD_0

USB2D USB Command Register

Offset: 0x130 | Read/Write: R/W | Reset: 0bxxxx000000001000000x1x11x0000000

Bit	R/W	Reset	Description
27:24	RW	0x0	HIRD: Host Initiated Resume Duration RW. Default = 0000b. This has the same behavior as bits 7:4 of the BA field of the HOSTPCx register. When writing to this field all BA[7:4] fields of all HOSTPCx registers will be set to this value. This field is used by system software to specify the minimum amount of time the host controller will drive the K-state during a host-initiated resume from an LPM state (e.g., L1), and is conveyed to each LPM-enabled device (via the HIRD bits within an LPM Tokens bmAttributes field) upon entry into a low-power state. Note the host controller is required to drive resume signaling for at least the amount of time specified in the HIRD value conveyed to the device during any proceeding host-initiated resume. Also note that the host controller is not required to observe this requirement during device-initiated resumes. Encoding for this field is identical to the definition for the similarly named HIRD field within an LPM Token, specifically: a value 0000b equals 50µs and each additional increment adds 75µs. For example, the value 0001b equals 125µs, and a value 1111b equals 1,175µs (~1.2ms).
23:16	RW	0x8	ITC: Interrupt Threshold Control .Read/Write. Default 08h. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are shown below. Value Maximum Interrupt Interval 00h Immediate (no threshold) 01h 1 micro-frame 02h 2 micro-frames 04h 4 micro-frames 08h 8 micro-frames 10h 16 micro-frames 20h 32 micro-frames 40h 64 micro-frames 0 = IMMEDIATE 2 = ONE_MF 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

Bit	R/W	Reset	Description
			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 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

Bit	R/W	Reset	Description
			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

19.13.4.19 USB2_CONTROLLER_1_USB2D_USBSTS_0

USB2D USB Status Register

Offset: 0x134 | Read/Write: R/W | Reset: 0b00000000000000000x100x0000x0000

Bit	R/W	Reset	Description
31:16	RW	0x0	PPCI: Port-n Change Detect - RW. Default = 0000h. The definition for each bit is identical to the Port Change Detect field (bit 2 of this register) except these bits are specific to a given port, where bit 16 = Port 1, 17 = Port 2, etc. For example, if bit 17 is set to a one then a port change event was detected on Port 2. The N_PORTS field in HCSPARAMS specifies how many ports are exposed by the host controller and thus how many bits in this field are valid.
15	RW	0x0	AS: Asynchronous Schedule Status. This bit reports the current status of the Asynchronous Schedule. When set to zero the asynchronous schedule status is disabled and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. If AS = ASE: 1= Enable Asynchronous Schedule. 0= Disable Asynchronous Schedule. Only used by the host controller. 0 = DISABLE 1 = ENABLE
14	RW	0x0	PS: Periodic Schedule Status. This bit reports the current status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. If PS = PSE:: 1 = Periodic Schedule is enabled. 0 = Periodic Schedule is disabled. Only used by the host controller. 0 = DISABLE 1 = ENABLE
13	RO	X	RCL: Reclamation. This is a read-only status bit used to detect an empty asynchronous schedule. Only used by the host controller. 0 = DISABLE 1 = ENABLE
12	RW	0x1	HCH: HCHalted. 1 = Default. This bit is a zero whenever the Run/Stop bit is a one. The Host Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g., internal error). Only used by the host controller. 0 = UNHALTED 1 = HALTED
11	RW	0x0	UALT_INT: ULPI alt_int Interrupt. 0 = Default. This interrupt bit is set when an RXCMD is received through the ULPI interface with bit 7 set (alt_int). The alt_int bit is set when an unmasked event occurs on any bit in the Carkit Interrupt Latch Register, in the ULPI PHY. The software should read the Carkit Interrupt Latch Register (Read to Clear) through the ULPI Viewport to check the source of the interrupt. Only present in designs where configuration constant VUSB_HS_PHY_ULPI = 1. 0 = NOT_ULPI_ALT_INT 1 = ULPI_ALT_INT
10	RW	0x0	ULPI_INT: ULPI Interrupt. This bit is set whenever an interrupt is received from ULPI PHY. Software writes 1 to clear it. 0 = NOT_ULPI_INT 1 = ULPI_INT

Bit	R/W	Reset	Description
8	RW	0x0	SLI: DCSuspend. When a device controller enters a suspend state from an active state, this bit will be set to a 1. The device controller clears the bit upon exiting from a suspend state. Only used by the device controller. 0 = NOTSUSPEND 1 = SUSPENDED
7	RW	0x0	SRI: SOF Received. When the device controller detects a Start Of (micro) Frame, this bit will be set to a one. When an SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1ms in device FS mode and every 125 µs in HS mode and will be synchronized to the actual SOF that is received. Since device controller is initialized to FS before connect, this bit will be set at an interval of 1ms during the prelude to the connect and chirp. In host mode, this bit will be set every 125 µs and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it. This is a non-EHCI status bit. 0 = SOF_NOT_RCVD 1 = SOF_RCVD
6	RW	0x0	URI: USB Reset Received. When the device controller detects a USB Reset and enters the default state, this bit is set to a 1. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used by the device controller. 0 = NO_USB_RESET 1 = USB_RESET
5	RW	0x0	AAI: Interrupt and Asynchronous Advance. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the USBCMD register. This status bit indicates the assertion of that interrupt source. Only used by the host controller. 0 = NOT_ADVANCED 1 = ADVANCED
4	RO	X	SEI: System Error. This bit is not used in this implementation and will always be set to "0". 0 = NO_ERROR 1 = ERROR
3	RW	0x0	FRI: Frame List Rollover. The Host Controller sets this bit to a 1 when the Frame List Index rolls over from its maximum value to 0. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the Frame List Size field of the USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [1 3] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a 1 every time FHINDEX [12] toggles. Only used by the host controller. 0 = NO_ROLLOVER 1 = ROLLOVER
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

19.13.4.20 USB2_CONTROLLER_1_USB2D_USBINTR_0

USB2D USB Interrupt Enable Register

Offset: 0x138 | Read/Write: R/W | Reset: 0b0000000000000000xxxx00x000000000

Bit	Reset	Description
31:16	0x0	PPCE: Port-n Change Detect Enable - RW. Default = 0000h. The definition for each bit in this field is identical to bit 2 of this register (Port Change Interrupt Enable) except these bits are specific to a given port, where bit 16 = Port 1, 17 = Port 2, etc. For example, if bit 17 is set (1b) then a port change event was detected on Port 2. When a bit in this field is a one, and the corresponding Port-n Change Detect bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Port-n Change Detect bit.
11	0x0	UALTIE: ULPI alt_int Interrupt Enable. 1 = USB controller issues an interrupt if the ULPI_ALT_INT bit in the USBSTS register transitions. The interrupt is acknowledged by software writing a 1 to the ULPI_ALT_INT bit. 0 = DISABLE 1 = ENABLE
10	0x0	ULPIE: ULPI Interrupt Enable. 1 = USB controller issues an interrupt if ULPI_INT bit in USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the ULPI_INT bit. 0 = DISABLE 1 = ENABLE
8	0x0	SLE: Sleep Enable. 1 = Device controller issues an interrupt if DCSuspend bit in USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the DCSuspend bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE
7	0x0	SRE: SOF Received Enable. 1 = Device controller issues an interrupt if SOF Received bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the SOF Received bit. 0 = DISABLE 1 = ENABLE
6	0x0	URE: USB Reset Enable.1 = Device controller issues an interrupt if USB Reset Received bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the USB Reset Received bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE
5	0x0	AAE: Interrupt on Asynchronous Advance Enable. 1 = the host controller issues an interrupt at the next interrupt threshold if the Interrupt on Async Advance bit in the USBSTS register = 1. The interrupt is acknowledged by software clearing the Interrupt on Async Advance bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
4	0x0	SEE: System Error Enable. 1 = Host/device controller issues an interrupt if the System Error bit in the USBSTS register = 1. The interrupt is acknowledged by software clearing the System Error bit. 0 = DISABLE 1 = ENABLE
3	0x0	FRE: Frame List Rollover Enable. 1 = Host controller issues an interrupt if the Frame List Rollover bit in the USBSTS register = 1. The interrupt is acknowledged by software clearing the Frame List Rollover bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
2	0x0	PCE: Port Change Detect Enable. 1 = Host/device controller issues an interrupt if Port Change Detect bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the Port Change Detect bit. 0 = DISABLE 1 = ENABLE
1	0x0	UEE: USB Error Interrupt Enable. 1 = Host controller issues an interrupt at the next interrupt threshold if the USBERRINT bit in USBSTS = 1. The interrupt is acknowledged by software clearing the USBERRINT bit in the USBSTS register. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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

19.13.4.21 USB2_CONTROLLER_1_USB2D_FRINDEX_0

USB2D USB Frame Index Register

Offset: 0x13c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
13:0	X	FRINDEX: Frame Index. The value in this register increments at the end of each time frame (micro-frame). Bits [N: 3] are used for the Frame List current index. Each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in host mode. USBCMD [Frame List Size] Number Elements N 000b (1024) 12 001b (512) 11 010b (256) 10 011b (128) 9 100b (64) 8 101b (32) 7 110b (16) 6 111b (8) 5 In device mode, the value is the current frame number of the last frame transmitted. It is not used as an index. In either mode, bits 2:0 indicate the current micro-frame.

19.13.4.22 USB2_CONTROLLER_1_USB2D_PERIODICLISTBASE_0

USB2D Host Controller Frame List Base Address Register

Offset: 0x144 | Read/Write: R/W | Reset: 0b00000000000000000000xxxxxxxxxxxx

Bit	Reset	Description
31:25	0x0	USBADR: Device mode. The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS request.
24	0x0	USBADRA: Device Address Advance. Default=0. When this bit is 0, any writes to USBADR are instantaneous. When this bit is written to a 1 at the same time or before USBADR is written, the write to the USBADR field is staged and held in a hidden register. After an IN occurs on endpoint 0 and is ACKed, USBADR will be loaded from the holding register. Hardware will automatically clear this bit on the following conditions: 1) IN is ACKed to endpoint 0. (USBADR is updated from staging register). 2) OUT/SETUP occur to endpoint 0. (USBADR is not updated). 3) Device Reset occurs (USBADR is reset to 0). Note: After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field. This mechanism will ensure this specification is met when the DCD cannot write of the device address within 2ms from the SET_ADDRESS status phase. If the DCD writes the USBADR with USBADRA=1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR will be programmed instantly at the correct time and meet the 2ms USB requirement.
31:12	0x0	BASEADR: Host mode: This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. HCD loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register are combined with the Frame Index Register (FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence. Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.

19.13.4.23 USB2_CONTROLLER_1_USB2D_ASYNCLISTADDR_0

USB2D Next Asynchronous List Address Register

Offset: 0x148 | Read/Write: R/W | Reset: 0b00000000000000000000000000xxxxx

Bit	Reset	Description
31:11	0x0	EPBASE: Device mode. This register contains the address of the top of the endpoint list in system memory. These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Heads (QH). Only used by the device controller.
31:5	0x0	ASYBASE: Host mode. This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (OH). Only used by the host controller.

19.13.4.24 USB2_CONTROLLER_1_USB2D_ASYNCTTSTS_0

USB2D Asynchronous Buffer Status for Embedded TT Register

Offset: 0x14c | Read/Write: R/W | Reset: 0bx0000000xxxxxxxxxxxxxxxxxxxxxx0x

Bit	R/W	Reset	Description
30:24	RW	0x0	TTHA: Internal TT Hub Address representation. This field is used to match the Hub Address field in QH (queue head) and siTD to determine if the packet is routed to the internal TT for directly attached FS/LS devices. If the Hub Address in the QH or siTD does not match this address, then the packet will be broadcast on the High-Speed ports destined for a downstream High Speed hub with the address in QH/siTD.
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.

19.13.4.25 USB2_CONTROLLER_1_USB2D_BURSTSIZE_0

USB2D Burst Size register

Offset: 0x150 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000100000001000

Bit	Reset	Description
15:8	0x8	TXPBURST: Programmable TX Burst Length. (Read/Write). This register represents the maximum length of a burst in 32-bit words while moving data from system memory to the USB bus.
7:0	0x8	RXPBURST: Programmable RX Burst Length. (Read/Write). This register represents the maximum length of a burst in 32-bit words while moving data from the USB bus to system memory.

19.13.4.26 USB2_CONTROLLER_1_USB2D_TXFILLTUNING_0

USB2D Transmit Fill Tuning Register

Offset: 0x154 | Read/Write: R/W | Reset: 0bxxxxxxxxxx000010xxx0000000000000

Bit	Reset	Description
21:16	0x2	TXFIFOTHRES: FIFO Burst Threshold. (Read/Write). This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2, and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO

Bit	Reset	Description
		may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USBMODE register is set.
12:8	0x0	TXSCHHEALTH: Scheduler Health Counter. (Read/Write To Clear) [Default = 0] This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max out at 31.
7:0	0x0	TXSCHOH: Scheduler Overhead. (Read/Write) [Default = 0] This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267 µs when a device is connected in High-Speed Mode. The time unit represented in this register is 6.333 µs when a device is connected in Low/Full Speed Mode

19.13.4.27 USB2_CONTROLLER_1_USB2D_ICUSB_CTRL_0

USB2D ICUSB Control Register

This register enables and controls the ICUSB FS/LS transceiver.

Offset: 0x15c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000

Bit	Reset	Description
3	0x0	IC_ENB1: ICUSB transceiver enable. This bit enables the ICUSB transceiver. To enable the interface, the PTS bits 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.

19.13.4.28 USB2_CONTROLLER_1_USB2D_ULPI_VIEWPORT_0

This register provides indirect access to the ULPI PHY register set. Although the USB controller performs access to the ULPI PHY register set, there may be extraordinary circumstances where software may need direct access.

Note: WRITES TO THE ULPI THROUGH THE VIEWPORT CAN SUBSTANTIALLY HARM STANDARD USB OPERATIONS. CURRENTLY NO USAGE MODEL HAS BEEN DEFINED WHERE SOFTWARE SHOULD NEED TO EXECUTE WRITES DIRECTLY TO THE ULPI PHY. SEE EXCEPTION REGARDING OPTIONAL FEATURES BELOW.

Note: EXECUTING READ OPERATIONS THROUGH THE ULPI VIEWPORT SHOULD HAVE NO HARMFUL SIDE EFFECTS TO STANDARD USB OPERATIONS.

There are two operations that can be performed with the ULPI Viewport-- wakeup and read /write operations.

The wakeup operation is used to put the ULPI interface into normal operation mode and re-enable the clock if necessary. A wakeup operation is required before accessing the registers when the ULPI interface is operating in low power mode, serial mode, or Carkit mode.

The ULPI state can be determined by reading the sync state bit (ULPI_SYNC_STATE). If this bit is a one, then ULPI interface is running in normal operation mode and can accept read/write operations. If the ULPI_SYNC_STATE indicates a 0 then then read/write operations will not be able to execute. Undefined behavior will result if ULPI_SYNC_STATE = 0 and a read or write operation is performed.

To execute a wakeup operation, write all 32 bits of the ULPI Viewport where ULPI_PORT is constructed appropriately and the ULPI_WAKEUP bit is a 1 and ULPI_RUN bit is a 0. Poll the ULPI Viewport until ULPI_WAKEUP is zero for the operation to complete.

To execute a read or write operation, write all 32 bits of the ULPI Viewport where ULPI_DATA_WR, ULPI_REG_ADDR, ULPI_PORT, ULPI_RD_WR are constructed appropriately and the ULPI_RUN bit is a 1. Poll the ULPI Viewport until ULPI_RUN is zero for the operation to complete. Once ULPI_RUN is zero, the ULPI_DATA_RD will be valid if the operation was a read.

The polling method above can be changed to interrupt driven using the ULPI interrupt defined in the USBSTS and USBINTR registers. When a wakeup or read/write operation is complete, the ULPI_INT interrupt will be set.

There are several optional features that may need to be enabled or disabled by system software as part of system configuration. These bits are contained in the Interface and OTG Control registers of the ULPI PHY register set. These registers also contain bits which are controlled by the link dynamically and therefore should be only modified by system software using the Set/Clear access method. Direct writes to these registers could have harmful side effects to the standard USB operations. The optional bits are as follows: Bits 3 through 7 in the Interface Control register and Bits 6 and 7 in the OTG Control register.

Please refer to the ULPI Specification Revision 1.1 for further information on the use of the optional features.

USB2D ULPI Viewport Register

Offset: 0x160 | Read/Write: R/W | Reset: 0b000xx00000000000xxxxxxxx00000000

Bit	R/W	Reset	Description
31	RW	0x0	ULPI_WAKEUP: ULPI Wakeup. Writing the 1 to this bit will begin the wakeup operation. The bit will automatically transition to 0 after the wakeup is complete. Once this bit is set, the driver cannot set it back to 0. Note: The driver must never execute a wakeup and a read/write operation at the same time. 0 = CLEAR 1 = SET
30	RW	0x0	ULPI_RUN: ULPI read/write Run. Writing the 1 to this bit will begin the read/write operation. The bit will automatically transition to 0 after the read/write is complete. Once this bit is set, the driver cannot set it back to 0. Note: The driver must never execute a wakeup and a read/write operation at the same time. 0 = CLEAR 1 = SET
29	RW	0x0	ULPI_RD_WR: ULPI read/write control. (0) Read; (1) Write. This bit selects between running a read or write operation. 0 = READ 1 = WRITE
27	RO	X	ULPI_SYNC_STATE: ULPI sync state. (1) Normal Sync. State. (0) In another state (i.e., Carkit, serial, low power). This bit represents the state of the ULPI interface. 0 = NOT_NORMAL 1 = NORMAL
26:24	RW	0x0	ULPI_PORT: ULPI PHY port number. This field should be always written as 0.
23:16	RW	0x0	ULPI_REG_ADDR: ULPI PHY register address. When doing a read or write operation to the ULPI PHY, the address of the ULPI PHY register being accessed is written to this field.
15:8	RO	X	ULPI_DATA_RD: ULPI PHY data read. The data from the ULPI PHY register can be read from here after the read operation completes.
7:0	RW	0x0	ULPI_DATA_WR: ULPI PHY data write. The data to write to the ULPI PHY register is written

Bit	R/W	Reset	Description
			here.

19.13.4.29 USB2_CONTROLLER_1_USB2D_PORTSC1_0

USB2D Port Status/Control 1 Register

Offset: 0x174 | Read/Write: R/W | Reset: 0b0000000xx0000000xxx1xx00x0xx010x

Bit	R/W	Reset	Description
31:25	RW	0x0	DA: Device Address. Default = 0000000b. The 7-bit USB device address for the device attached to an immediately downstream of the associated root port. A value of zero indicates no device is present or software support for this feature is not present. This is used by the Controller when sending the LPM token. This field is only valid when the core is operating in host mode. If in device mode it will be read only and always equal to 0000000b.
24:23	RO	X	SSTS: Suspend Status. Default = 00b. These two bits are used by software to determine whether an L1-based suspend request was successful, specifically: 00b - L1 state entered with success. ACK received from peripheral. 01b - NYET received from peripheral. It was not able to enter L1 state this time. 10b - L1 state not supported by peripheral. STALL received. 11b - Peripheral did not respond or an error occurred. The value of this field is only valid when the port resides in the L0 state - that is, the meaning of these bits is invalid whenever bit 7 of this register (SUSP) is one. Ideally, the Controller driver should read this register if it receives an interrupt after issuing a suspend using L1 support. In case of a non-success a port change interrupt will be fired and this field should be checked for a possible L1 failure. This field is only valid when the core is operating in host mode. If in device mode it will be always equal to 00b. 0 = L1STATE_ENTERED 1 = NYET_PERIPH 2 = L1STATE_NOT_SUPPORTED 3 = PERIPH_NORESP_ERR
22	RW	0x0	WKOC: Default = 0b. Wake on Over-current Enable: Writing this bit to a one enables the port to be sensitive to over-current conditions as wake-up events. This field is zero if Port Power (PP) is zero. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined behavior. 0 = DISABLE 1 = ENABLE
21	RW	0x0	WKDS: Wake on Disconnect Enable: Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This field is zero if Port Power (PP) is zero or in device mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined behavior. This bit should not be written to 1 if there is no device connected. After the device disconnect is detected, this bit should be cleared to 0. 0 = DISABLE 1 = ENABLE
20	RW	0x0	WKCN: Wake on Connect Enable: Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This field is zero if Port Power (PP) is zero or in device mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined behavior. This bit should not be written to 1 while the device is connected. After the device connection is detected, this bit should be cleared to 0. 0 = DISABLE 1 = ENABLE
19:16	RW	0x0	PTC: Port Test Control: Any value other than zero indicates that the port is operating in test mode. Value Specific Test. 0000b Not enabled. 0001b J_ STATE. 0010b K_STATE.

Bit	R/W	Reset	Description
			0011b SEQ_NAK. 0100b Packet. 0101b FORCE_ENABLE. 0110b to 1111b Reserved. Refer to Chapter 7 of the USB Specification Revision 2.0 for details on each test mode. 0 = NORMAL_OP 1 = TEST_J 2 = TEST_K 3 = TEST_SE0_NAK 4 = TEST_PKT 5 = TEST_FORCE_ENABLE
15:14	RO	X	PIC: Port Indicator Control: This field is not supported in the current implementation. Please use a GPIO if you wish to use Port Indicators.
13	RO	X	PO: Port Owner. Port owner handoff is not implemented in this design, therefore this bit will always be 0.
12	RW	0x1	PP: Port Power: The function of this bit depends on the value of the Port Power Switching (PPC) field in the HCSPARAMS register. The behavior is as follows: PPC PP Operation 0b 0b Read Only. A device controller with no OTG capability does not have port power control switches. 1b 1b/0b RW. Host/OTG controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (i.e., PP equals a 0), the port is non-functional and will not report attaches, detaches, etc. When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitioned by the host controller driver from a one to a zero (removing power from the port). 0 = NOT_POWERED 1 = POWERED
11:10	RO	X	LS: These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. The encoding of the bits is: 00b = SE0 10b = J-state 01b = 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 = K_STATE 2 = J_STATE 3 = UNDEFINED
9	RW	0x0	SLP: Suspend using L1 - RW. Default = 0b. When this bit is set to '1' and a non-zero Device Address (DA) is specified Controller will instigate L1 entry during suspend (bit 7) and L1 exit during resume (bit 6). When set to zero the Controller will use the legacy (L2) mechanism. Software should only set this bit when the device attached immediately downstream supports L1 transitions. When acting as device, this bit is read-only and set to '1' by the hardware when the Controller enters is L1 state (LPM token received and accepted). Note: HSP is redundant with PSPD(27:26). This bit is not defined in the EHCI specification.
8	RW	0x0	PR: This field is zero if Port Power (PP) is zero. In Host Mode: Read/Write. 1=Port is in Reset. 0=Port is not in Reset. When software writes a one to this bit, the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver. In Device Mode: This bit is a read-only status bit. Device reset from the USB bus is also indicated in the USBSTS register. 0 = NOT_USB_RESET 1 = USB_RESET

Bit	R/W	Reset	Description
7	RO	X	SUSP: Port suspend. 1=Port in suspend state. 0=Port not in suspend state. In Host Mode: Read/Write. Port Enabled bit and Suspend bit of this register define the port states as follows: Bits [Port Enabled, Suspend] Port State 0x Disable 10 Enable 11 Suspend When in the suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to zero when software sets the Force Port Resume bit to zero. A write of zero to this bit is ignored by the host controller. If host software sets this bit to a one when the port is not enabled (i.e., Port enabled bit is a zero), the results are undefined. This field is zero if Port Power (PP) is zero in host mode. In Device Mode: Read Only. This bit is a read-only status bit. 0 = NOT_SUSPEND 1 = SUSPEND
6	RW	0x0	FPR: Force Port Resume. 1= Resume detected/driven on port. 0=No resume (K state) detected/driven on port. In Host Mode: Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver. Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. This bit remains a one until the port has switched to the high-speed idle. Writing a zero has no effect because the port controller will time the resume operation to clear the bit when the port control state switches to HS or FS idle. This field is zero if Port Power (PP) is zero in host mode. This bit is not-EHCI compatible. In Device mode: After the device has been in Suspend State for 5 ms or more, software must set this bit to one to drive resume signaling before clearing. The Device Controller will set this bit to one if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. Software should ensure that the PHY clock is operational before writing a 1 to this bit to start the resume sequence. This is true for both Device and Host modes. 0 = NO_RESUME 1 = RESUME
5	RO	X	OCC: Over-current Change: Not supported 0 = NO_CHANGE 1 = CHANGE
4	RO	X	OCA: Over-current Active: Not supported 0 = NO_OVER_CURRENT 1 = OVER_CURRENT
3	RW	0x0	PEC: Port Enable/Disable Change: 1=Port enabled/disabled status has changed. 0=No change. In Host Mode: For the root hub, this bit gets set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a one to it. This field is zero if Port Power (PP) is zero. In Device mode: The device port is always enabled. (This bit will be zero) 0 = NO_CHANGE 1 = CHANGE
2	RW	0x1	PE: Port Enabled/Disabled: 1=Enable. 0=Disable (default) In Host Mode: Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0b)

Bit	R/W	Reset	Description
			downstream propagation of data is blocked except for reset. This field is zero if Port Power (PP) is zero in host mode. In Device Mode: The device port is always enabled. (This bit will be one) 0 = PORT_DISABLED 1 = PORT_ENABLED
1	RW	0x0	CSC: Connect Status Change: 1 =Change in Current Connect Status. 0=No change (default) In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (i.e., the bit will remain set). Software clears this bit by writing a one to it. This field is zero if Port Power (PP) is zero in host mode. This bit is undefined in device controller mode. 0 = NO_CHANGE 1 = CHANGE
0	RO	X	CCS: Current Connect Status. In Host Mode: 1=Device is present on port. 0=No device is present (default). This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set. This field is zero if Port Power (PP) is zero in host mode. In Device Mode: 1=Attached. 0=Not Attached (default). A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended. 0 = NOT_CONNECTED 1 = CONNECTED

19.13.4.30 USB2_CONTROLLER_1_USB2D_HOSTPC1_DEVLC_0

USB2D Device Mode LPM Behavior and Control Register

This register controls the LPM behavior and controls the behavior of each USB port. Only available 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 Behavior and Control Register

This register controls the LPM behavior and controls the behavior of each USB port. Only available when in host mode. This register shares the same address with the DEVLC register (which is used only in device mode).

Offset: 0x1b4 | Read/Write: R/W | Reset: 0b0000xxx0000000000000xxxxxxxxxxx0

Bit	R/W	Reset	Description
31:29	RW	0x0	PTS: Parallel transceiver select. This bit is not defined in the EHCI specification. 0 = UTMI 1 = RESERVED 2 = ULPI 3 = ICUSB_SER
28	RW	0x0	STS: Serial transceiver not selected. This is the only value supported. This bit is not defined in the EHCI specification. 0 = PARALLEL_IF 1 = SERIAL_IF
27	RO	X	PTW: Parallel Transceiver Width. Fixed to 0. This bit is not defined in the EHCI specification. 0 = EIGHT_BIT 1 = RESERVED

Bit	R/W	Reset	Description
26:25	RO	X	PSPD: This register field indicates the speed at which the port is operating. 00 = Full Speed 01 = Low Speed 10 = High Speed. This bit is not defined in the EHCI specification. 0 = FULL_SPEED 1 = LOW_SPEED 2 = HIGH_SPEED 3 = RESERVED
24	RW	0x0	ALPD: Auto Low Power While Disconnect - RW. Default = 0b. If set, this feature will be enabled, and every time the port enters the disconnect state, it will also enter in low power state, disabling the transceiver clock. The behavior will be same as if the PHCD bit was enabled (in fact this bit will be set to 1 as soon as low power mode is enabled).When this field is set the WKCN field of PORTSCx register (Wake on Connect Enable) will also be set. This way the core will wake up in case a connect is detected. There will be a delay between the detection of a disconnect and actually enter in low power mode. This delay can be controlled by writing to the ALPDD field in the register USBMODE 0 = DONT_AUTO_LOW_POWER_WHILE_DISCONNECT 1 = AUTO_LOW_POWER_WHILE_DISCONNECT
23	RW	0x0	PFSC: Port Force Full Speed Connect - RW. Default = 0b. Writing this bit to a '1b' will force the port to only connect at Full Speed. It disables the chirp sequence that allows the port to identify itself as High Speed. This is useful for testing FS configurations with an HS host, hub or device. This bit is not defined in the EHCI specification. 0 = DONT_FORCE_FULL_SPEED 1 = FORCE_FULL_SPEED
22	RW	0x0	PHCD: PHY Low Power Suspend - Clock disable: Writing this bit to a 1 will disable the PHY clock. Writing a 0 enables it. Reading this bit will indicate the status of the PHY clock. NOTE: The PHY clock cannot be disabled if it is being used as the system clock. In device mode, the PHY can be put into Low Power Clock Disable when the device is not running (USBCMD RS=0b) or the host has signaled suspend (PORTSCx SUSP=1b). Low Power Clock Disable will be cleared automatically when the host has signaled resume. Before forcing a resume from the device, the Controller driver must clear this bit. In host mode, the PHY can be put into Low Power Suspend Clock Disable when the downstream device has been put into suspend mode or when no downstream device is connected. Low Power Clock Disable is completely under the control of software. 0 = DISABLE 1 = ENABLE
21:20	RW	0x0	LPMX: Auto LPM set - RW. Default = 00b. This bit field is valid during Host Mode Only. For Device Mode this is reserved. Value Meaning 00b Disables auto LPM. 01b If there is no activity for a certain number of SOFs the controller will send an LPM token, enter in suspend and issue a port change interrupt. 10b Same as above but without issuing an interrupt. 11b Reserved for futures LPM enhancements. The detection of no activity is based on the absence of response from the downstream device. Because of this limitation this feature should not be used if ISO OUT endpoints are being used (as no handshake is expected).
17	RW	0x0	ASUS: Auto Low Power - RW. Default = 0b. This bit field is valid during Device Mode Only. In Host Mode it is part of ELPM field. This bit is used to control the auto low power feature. If set, the auto low power feature will be enabled and every time the port enters in suspend state it will also enter in low power state, disabling the transceiver clock. The behavior will be the same as if the PHCD bit was enabled (in fact this bit will be set to '1' as soon as low power mode is enabled). 0 = DISABLE 1 = ENABLE
19:16	RW	0x0	EPLPM: Endpoint for LPM token - RW. Default = 0000b. This bit field is valid during Host Mode only. For Device Mode [19:18] is reserved and 17 is ASUS, while 16 is STL. This sets the endpoint number to which the LPM token will be sent. It will be directly mapped to the ENDP field in the LPM token with the EXT PID.
16	RW	0x0	STL: STALL reply to LPM token - RW. Default = 0b. This bit field is valid during Device Mode Only. In Host Mode it is a part of ELPM field. When this bit is set to '1', the Controller will reply always with STALL to all incoming LPM tokens. This bit overrides the LPM NYET bit (NYT).

Bit	R/W	Reset	Description
			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 SOFs with no activity in between reaches this threshold and the auto LPM is enabled, an LPM token is sent and the port enters the suspend state. The SOF counter for this threshold is incremented each 125 µs, even if the port is not in HS operation.
11:1	RO	X	BA: bmAttributes - RO. Default = 00000000000b. This holds the bmAttributes field of the LPM sub-token received, after the EXT PID token.
0	RW	0x0	NYT_ASUS: NYET reply to LPM token - RW. Default = 0b. This bit is NYT during Device Mode. When this bit is '1', the device controller will NYET all the LPM tokens. When this bit is set to '0', the Controller will ACK all the LPM tokens if the STALL bit (STL) is also set to '0'. This bit is ASUS in Host Mode. This bit is used to control the auto low power feature. If set, the auto low power feature is enabled and every time the port enters the suspend state, it also enters the low power state, disabling the transceiver clock. The behavior is the same as if the PHCD bit was enabled (this bit is set to '1' as soon as low power mode is enabled). 0 = DISABLE 1 = ENABLE

19.13.4.31 USB2_CONTROLLER_1_USB2D_OTGSC_0

USB2D On-The-Go (OTG) Status and Control Register

Offset: 0x1f4 | Read/Write: R/W | Reset: 0bx0000000x0000000xxxxxxxxxx100x00

Bit	R/W	Reset	Description
30	RW	0x0	DPIE: Data Pulse Interrupt Enable. Setting this bit enables the Data pulse interrupt. 0 = DISABLE 1 = ENABLE
29	RW	0x0	ONEMSE: 1 millisecond timer Interrupt enable. Setting this bit enables the 1 millisecond timer interrupt. 0 = DISABLE 1 = ENABLE
28	RW	0x0	BSEIE: B Session End Interrupt Enable. Setting this bit enables the B session end interrupt 0 = DISABLE 1 = ENABLE
27	RW	0x0	BSVIE: B Session Valid Interrupt Enable. Setting this bit enables the B session valid interrupt 0 = DISABLE 1 = ENABLE
26	RW	0x0	ASVIE: A Session Valid Interrupt Enable. Setting this bit enables the A session valid interrupt 0 = DISABLE 1 = ENABLE
25	RW	0x0	AVVIE: A VBus Valid Interrupt Enable. Setting this bit enables the A VBus valid interrupt 0 = DISABLE 1 = ENABLE
24	RW	0x0	IDIE: USB ID Interrupt Enable. Setting this bit enables the USB ID interrupt 0 = DISABLE 1 = ENABLE
22	RW	0x0	DPIS: Data Pulse Interrupt Status. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USBMODE.CM = Host (11) and PORTSC(0). PortPower = Off (0). Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
21	RW	0x0	ONEMSS: 1 millisecond timer Interrupt Status: This bit is set once every millisecond. Software writes a 1 to clear it. 0 = INT_CLEAR 1 = INT_SET

Bit	R/W	Reset	Description
20	RW	0x0	BSEIS: B Session End Interrupt Status. This bit is set when VBus has fallen below the B session end threshold. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
19	RW	0x0	BSVIS: B Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the B session valid threshold (0.8 VDC). Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
18	RW	0x0	ASVIS: A Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the A session valid threshold (0.8 VDC). Software writes a one to clear this bit. 0 = INT_CLEAR 1 = INT_SET
17	RW	0x0	AVVIS: A VBus Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the VBus valid threshold (4.4 VDC) on an A device. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
16	RW	0x0	IDIS: USB ID Interrupt Status. This bit is set when a change on the ID input has been detected. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
14	RO	X	DPS: Data Bus Pulsing Status. A 1 indicates data bus pulsing is being detected on the port. 0 = STS_CLEAR 1 = STS_SET
13	RO	X	ONEMST: 1 millisecond timer toggle. This bit toggles once per millisecond 0 = STS_CLEAR 1 = STS_SET
12	RO	X	BSE: B session End. Indicates VBus is below the B session end threshold 0 = STS_CLEAR 1 = STS_SET
11	RO	X	BSV: B Session Valid. Indicates VBus is above the B session valid threshold 0 = STS_CLEAR 1 = STS_SET
10	RO	X	ASV: A Session Valid. Indicates VBus is above the A session valid threshold 0 = STS_CLEAR 1 = STS_SET
9	RO	X	AVV: A VBus Valid. Indicates VBus is above the A VBus valid threshold 0 = STS_CLEAR 1 = STS_SET
8	RO	X	ID: USB ID: 0 = A-device 1 = B-device 0 = A_DEV 1 = B_DEV
5	RW	0x1	IDPU: USB ID Pullup 0 = CLEAR 1 = SET
4	RW	0x0	DP: Data Pulsing. Setting this bit causes the pull-up on DP to be asserted for data pulsing during SRP. 0 = NO_DATA_PULSE 1 = DATA_PULSE
3	RW	0x0	OT: OTG Termination. This bit must be set when the OTG device is in device mode, this controls the pulldown on DM. 0 = NO_OTG_TERM 1 = OTG_TERM
1	RW	0x0	VC: VBUS Charge. Setting this bit causes the VBus line to be charged. This is used for VBus pulsing during SRP. 0 = NO_VBUS_CHRG 1 = VBUS_CHRG

Bit	R/W	Reset	Description
0	RW	0x0	VD: VBUS_Discharge. Read/write. Setting this bit causes Vbus to discharge through a resistor. 0 = NO_VBUS_DISCHRG 1 = VBUS_DISCHRG

19.13.4.32 USB2_CONTROLLER_1_USB2D_USBMODE_0

USB2D USB Device Mode Register

Offset: 0x1f8 | Read/Write: R/W | Reset: 0b00000000000000000xxxxxxxx0xxxxx

Bit	R/W	Reset	Description
31:16	RW	0x0	ALPDD: Auto Low Power While Disconnect Delay. Used when the ALPD field of the HOSTPCx register is set. Defines the delay between disconnect detection and entering in low power mode. The delay is <this register value>*64*100/3 in milliseconds. The maximum value is 65535*64*100/3 = 139,808 ms. The minimum value is 0. Only used in Host mode.
15	RW	0x0	SRT: Shorten USB Reset Time. Software should never set this to 1.
5	RW	0x0	VBPS: VBUS Power Select This can be used by logic that selects between an on-chip Vbus power source (charge pump) and an off-chip source in systems when both are available. Only to be used in Host Mode. Software should not use this.
4	RO	X	SDIS: Stream disable: 1 Streaming is disabled - helpful to avoid overruns/underruns when system load is too high. 0 = STREAM_ENABLE 1 = STREAM_DISABLE
3	RO	X	SLOM: Setup Lockout Mode: In device mode, this bit controls the behavior of the setup lockout mechanism. 0 - Setup lockout is ON (default) 1 Setup lockout is OFF. Firmware requires the use of setup tripwire semaphore in USB2D_USBCMD register. 0 = LOCKOUT_ON 1 = LOCKOUT_OFF
2	RO	X	ES: Endian Select: Note: For this implementation, this should be always set to 0 (little endian). 0 = LITTLE_ENDIAN 1 = RESERVED
1:0	RO	X	CM: Controller Mode: The controller mode will default to an idle state and will need to be initialized to the desired operating mode after reset. This register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the RESET bit in the USBCMD register before reprogramming this register. 00 = Idle [Default]. 01 = Reserved. 10 = Device Controller. 11 = Host Controller. 0 = IDLE 1 = RESERVED 2 = DEVICE_MODE 3 = HOST_MODE

19.13.4.33 USB2_CONTROLLER_1_USB2D_ENDPTNAK_0

USB2D Endpoint NAK register

Offset: 0x200 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	EPTN: TX Endpoint NAK R/WC. Each TX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received IN token for the corresponding endpoint. 0 = CLEAR 1 = SET
15:0	0x0	EPRN: RX Endpoint NAK R/WC. Each RX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received OUT or PING token for the corresponding endpoint. 0 = CLEAR 1 = SET

19.13.4.34 USB2_CONTROLLER_1_USB2D_ENDPTNAK_ENABLE_0

USB2D Endpoint NAK Enable Register

Offset: 0x204 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	EPTNE: TX Endpoint NAK Enable R/W. Each bit is an enable bit for the corresponding TX Endpoint NAK bit. If this bit is set and the corresponding TX Endpoint NAK bit is set, the NAK Interrupt bit is set. 0 = DISABLE 1 = ENABLE
15:0	0x0	EPRNE: RX Endpoint NAK Enable R/W. Each bit is an enable bit for the corresponding RX Endpoint NAK bit. If this bit is set and the corresponding RX Endpoint NAK bit is set, the NAK Interrupt bit is set. 0 = DISABLE 1 = ENABLE

19.13.4.35 USB2_CONTROLLER_1_USB2D_ENDPTSETUPSTAT_0

USB2D Endpoint Setup Status Register

Offset: 0x208 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
15	0x0	ENDPTSETUPSTAT15: Endpoint 15 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
14	0x0	ENDPTSETUPSTAT14: Endpoint 14 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
13	0x0	ENDPTSETUPSTAT13: Endpoint 13 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
12	0x0	ENDPTSETUPSTAT12: Endpoint 12 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
11	0x0	ENDPTSETUPSTAT11: Endpoint 11 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
10	0x0	ENDPTSETUPSTAT10: Endpoint 10 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total

Bit	Reset	Description
		response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
9	0x0	ENDPTSETUPSTAT9: Endpoint 9 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
8	0x0	ENDPTSETUPSTAT8: Endpoint 8 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
7	0x0	ENDPTSETUPSTAT7: Endpoint 7 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
6	0x0	ENDPTSETUPSTAT6: Endpoint 6 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
5	0x0	ENDPTSETUPSTAT5: Endpoint 5 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
4	0x0	ENDPTSETUPSTAT4: Endpoint 4 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
3	0x0	ENDPTSETUPSTAT3: Endpoint 3 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
2	0x0	ENDPTSETUPSTAT2: Endpoint 2 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD

Bit	Reset	Description
1	0x0	ENDPTSETUPSTAT1: Endpoint 1 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
0	0x0	ENDPTSETUPSTAT0: Endpoint 0 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD

19.13.4.36 USB2_CONTROLLER_1_USB2D_ENDPTPRIME_0

USB2D Endpoint Initialization Register

Offset: 0x20c | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	PETB15: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
30	0x0	PETB14: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
29	0x0	PETB13: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
28	0x0	PETB12: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
27	0x0	PETB11: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
26	0x0	PETB10: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
25	0x0	PETB9: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
24	0x0	PETB8: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
23	0x0	PETB7: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
22	0x0	PETB6: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
21	0x0	PETB5: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
20	0x0	PETB4: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
19	0x0	PETB3: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
18	0x0	PETB2: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
17	0x0	PETB1: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
16	0x0	PETB0: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
15	0x0	PERB15: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
14	0x0	PERB14: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
13	0x0	PERB13: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
12	0x0	PERB12: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
11	0x0	PERB11: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
10	0x0	PERB10: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
9	0x0	PERB9: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
8	0x0	PERB8: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
7	0x0	PERB7: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
6	0x0	PERB6: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
5	0x0	PERB5: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
4	0x0	PERB4: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
3	0x0	PERB3: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
2	0x0	PERB2: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
1	0x0	PERB1: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
0	0x0	PERB0: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

19.13.4.37 USB2_CONTROLLER_1_USB2D_ENDPTFLUSH_0

USB2D Endpoint De-Initialization Register

Offset: 0x210 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	FETB15: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
30	0x0	FETB14: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
29	0x0	FETB13: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
28	0x0	FETB12: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
27	0x0	FETB11: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
26	0x0	FETB10: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
25	0x0	FETB9: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
24	0x0	FETB8: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
23	0x0	FETB7: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
22	0x0	FETB6: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
21	0x0	FETB5: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
20	0x0	FETB4: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
19	0x0	FETB3: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
18	0x0	FETB2: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
17	0x0	FETB1: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
16	0x0	FETB0: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
15	0x0	FERB15: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
14	0x0	FERB14: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
13	0x0	FERB13: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
12	0x0	FERB12: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
11	0x0	FERB11: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
10	0x0	FERB10: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
9	0x0	FERB9: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
8	0x0	FERB8: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
7	0x0	FERB7: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
6	0x0	FERB6: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
5	0x0	FERB5: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
4	0x0	FERB4: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
3	0x0	FERB3: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
2	0x0	FERB2: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
1	0x0	FERB1: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
0	0x0	FERB0: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

19.13.4.38 USB2_CONTROLLER_1_USB2D_ENDPTSTATUS_0

USB2D Endpoint Status Register

Offset: 0x214 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	ETBR15: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
30	X	ETBR14: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
29	X	ETBR13: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
28	X	ETBR12: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
27	X	ETBR11: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
26	X	ETBR10: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
25	X	ETBR9: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
24	X	ETBR8: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
23	X	ETBR7: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
22	X	ETBR6: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
21	X	ETBR5: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
20	X	ETBR4: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
19	X	ETBR3: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
18	X	ETBR2: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
17	X	ETBR1: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
16	X	ETBR0: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
15	X	ERBR15: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
14	X	ERBR14: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
13	X	ERBR13: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
12	X	ERBR12: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
11	X	ERBR11: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
10	X	ERBR10: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
9	X	ERBR9: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
8	X	ERBR8: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
7	X	ERBR7: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
6	X	ERBR6: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
5	X	ERBR5: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
4	X	ERBR4: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
3	X	ERBR3: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
2	X	ERBR2: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
1	X	ERBR1: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
0	X	ERBR0: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

19.13.4.39 USB2_CONTROLLER_1_USB2D_ENDPTCOMPLETE_0

USB2D Endpoint Complete Register

Offset: 0x218 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ETCE15: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
30	0x0	ETCE14: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
29	0x0	ETCE13: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
28	0x0	ETCE12: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
27	0x0	ETCE11: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
26	0x0	ETCE10: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
25	0x0	ETCE9: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
24	0x0	ETCE8: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
23	0x0	ETCE7: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
22	0x0	ETCE6: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
21	0x0	ETCE5: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
20	0x0	ETCE4: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
19	0x0	ETCE3: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
18	0x0	ETCE2: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
17	0x0	ETCE1: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
16	0x0	ETCE0: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
15	0x0	ERCE15: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
14	0x0	ERCE14: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
13	0x0	ERCE13: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
12	0x0	ERCE12: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
11	0x0	ERCE11: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
10	0x0	ERCE10: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
9	0x0	ERCE9: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
8	0x0	ERCE8: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
7	0x0	ERCE7: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
6	0x0	ERCE6: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
5	0x0	ERCE5: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
4	0x0	ERCE4: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
3	0x0	ERCE3: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
2	0x0	ERCE2: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
1	0x0	ERCE1: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
0	0x0	ERCE0: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

19.13.4.40 USB2_CONTROLLER_1_USB2D_ENDPTCTRL0_0

USB2D Endpoint Control 0 Register

Offset: 0x21c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23	X	TXE: TX Endpoint Enable. Endpoint 0 is always enabled. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
19:18	X	TXT: TX Endpoint Type. Endpoint 0 is fixed as a Control Endpoint. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
16	X	TXS: TX Endpoint Stall: Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0 = EP_OK 1 = EP_STALL
7	X	RXE: RX Endpoint Enable. Endpoint 0 is always enabled. 0 = DISABLE 1 = ENABLE
3:2	X	RXT: RX Endpoint Type. Endpoint 0 is fixed as a Control Endpoint. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
0	X	RXS: RX Endpoint Stall: Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0 = EP_OK 1 = EP_STALL

19.13.4.41 USB2_CONTROLLER_1_USB2D_ENDPTCTRLn_0

USB2D Endpoint Control n Register

USB2D Endpoint Control 1 through 15 registers have the same field definitions and reset value. In the register offset, the value n is the register number (1 through 15).

Offset: 0x220 + (n – 1) * 0x04 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above.

Bit	R/W	Reset	Description
			0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above, 0 = EP_OK 1 = EP_STALL

19.13.5 USB2 Controller Interface Registers

These are used to generate the actual RTL registers for USB2 controller interface.

ULPIS2S resets to be used as follows:

- Assert ULPIS2S_SLV0_CLAMP_XMIT to ensure that the signals between SLV0 and the line simulator are clean.
- Assert ULPIS2S_SLV0_RESET
- Do a functional asynchronous reset of the ChipIdea controller
- Deassert ULPIS2S_SLV0_RESET
- Deassert ULPIS2S_SLV0_CLAMP_XMIT

NULPI_SLV1_RESET should be used in the same way if there is some explicit way to reset the external ULPI controller (which is not necessarily the case). There is usually no reason to assert ULPIS2S_LINE_RESET.

19.13.5.1 USB2_IF_USB_SUSP_CTRL_0

This register controls the suspend and resume behavior of the USB controller/PHY.

USB Suspend Control Register

Offset: 0x400 | Read/Write: R/W | Reset: 0bxxxxx00111110000x1001000xx000000

Bit	R/W	Reset	Description
26	RW	0x0	FAST_WAKEUP_RESP: Enable Fast Response from UTMIP PHY for a Remote Wakeup request from device. This is used only for cases where wakeup response needs to be within 1ms of spec. Used for Host mode ONLY.

Bit	R/W	Reset	Description
			0 = DISABLE 1 = ENABLE
25	RW	0x0	UTMIP_SUSPL1_SET: Enable SuspendL1 for UTMIP PHY Enabling this will only cutoff clocks to the UTMIP logic. The USB PLLs, PllU and UTMIP PLL will still be running. 0 = DISABLE 1 = ENABLE
24	RW	0x1	ULPI_PADS_CLKEN_RESET: Async reset for the synchronizers that are used in the external and loopback ULPI 60 MHz clock. 0 = DISABLE 1 = ENABLE
23	RW	0x1	ULPI_PADS_RESET: Async reset for trimmers and line state logic that is implemented in the pad macros. 0 = DISABLE 1 = ENABLE
22	RW	0x1	ULPIS2S_LINE_RESET: Async reset of the line simulator logic that sits between the two virtual PHYs (active high). The ULPI PHY registers should be programmed while ULPI is in reset. 0 = DISABLE 1 = ENABLE
21	RW	0x1	ULPIS2S_SLV1_RESET: Async reset of the SLV1 ULPI logic. This corresponds to resetting the virtual PHY that is connected to the internal ULPI controller (active high). The ULPI PHY registers should be programmed while ULPI is in reset. 0 = DISABLE 1 = ENABLE
20	RW	0x1	ULPIS2S_SLV0_RESET: Async reset of the SLV0 ULPI logic. This corresponds to resetting the virtual PHY that is connected to the external ULPI controller. (active high). The ULPI PHY registers should be programmed while ULPI is in reset. 0 = DISABLE 1 = ENABLE
19	RW	0x0	UHSIC_PHY_ENB: Enable UHSIC PHY mode. Set this to 1 if using UHSIC PHY. Otherwise set this to 0. 0 = DISABLE 1 = ENABLE
18:16	RW	0x0	USB_WAKEUP_DEBOUNCE_COUNT: USB PHY wakeup debounce counter USB will debounce any wakeup event by the number of clocks programmed in this counter. A value of 0 results in no debounce.
14	RW	0x1	UHSIC_RESET: Reset going to UHSIC PHY (active high). This should be set to 1 whenever programming the UHSIC config registers. It should be cleared to 0 after the programming of UHSIC config registers is done. UHSIC config registers should be programmed only once before doing any transactions on UHSIC. The UHSIC PHY registers should be programmed while UHSIC is in reset. 0 = DISABLE 1 = ENABLE
13	RW	0x0	ULPI_PHY_ENB: Enable ULPI PHY mode. Set this to 1 if using null or link ULPI PHY. Otherwise set this to 0. 0 = DISABLE 1 = ENABLE
12	RW	0x0	UTMIP_PHY_ENB: Enable UTMIP PHY mode Set this to 1 if using UTMIP PHY. Otherwise set this to 0. 0 = DISABLE 1 = ENABLE
11	RW	0x1	UTMIP_RESET: Reset going to UTMIP PHY (active high). This should be set to 1 whenever programming the UTMIP config registers. It should be cleared to 0 after the programming of UTMIP config registers is done. UTMIP config registers should be programmed only once before doing any transactions on UTMIP. The UTMIP PHY registers should be programmed while UTMIP is in reset. 0 = DISABLE 1 = ENABLE
10	RW	0x0	USB_SUSP_POL: Polarity of the suspend signal going to UHSIC PHY.

Bit	R/W	Reset	Description
			0 = Active low (default) 1 = Active high. This should not be changed by software. 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
9	RW	0x0	USB_PHY_CLK_VALID_INT_ENB: USB PHY clock valid interrupt enable. If this bit is enabled, interrupt is generated whenever USB clocks are resumed from a suspend. 0 = DISABLE 1 = ENABLE
8	RO	0x0	USB_PHY_CLK_VALID_INT_STS: USB PHY clock valid interrupt status. This bit is set whenever USB PHY clock is woken up from suspend. Software must write a 1 to clear this bit. 0 = UNSET 1 = SET
7	RO	X	USB_PHY_CLK_VALID: USB PHY clock valid status. This bit indicates whether the USB PHY is generating a valid clock to the USB controller. If USB PHY clock is running, this bit is set to 1, else it is set to 0. 0 = UNSET 1 = SET
6	RO	X	USB_CLKEN: USB AHB clock enable status. Indicates whether the AHB clock to the USB controller is enabled or not. If AHB clock to USB controller is enabled, this bit is set to 1, else it is set to 0. NOTE: even when this is set to 0, all essential blocks that are required to resume USB clocks from suspend will be active and their AHB clock will not be suspended. 0 = UNSET 1 = SET
5	RW	0x0	USB_SUSP_CLR: Suspend Clear Software must write a 1 to this bit to bring the PHY out of suspend mode. This is used when the software stops the PHY clock during suspend and then wants to initiate a resume. Software should also write 0 to clear it. NOTE: It is required that software generate a positive pulse on this bit to guarantee proper operation. 0 = UNSET 1 = SET
4	RW	0x0	USB_WAKE_ON_DISCON_EN_DEV: Wake on Disconnect Enable (device mode). When enabled (1), USB device will wake up from suspend on a disconnect event. This is only valid when USB controller is in device mode, it is not applicable when USB controller is in host mode. 0 = DISABLE 1 = ENABLE
3	RW	0x0	USB_WAKE_ON_CNNT_EN_DEV: Wake on Connect Enable (device mode). When enabled (1), USB device will wake up from suspend on a connect event. This is only valid when USB controller is in device mode, it is not applicable when USB controller is in host mode. 0 = DISABLE 1 = ENABLE
2	RW	0x0	USB_WAKE_ON_RESUME_EN: Wake on resume enable. If this bit is enabled, the USB will wake up from suspend whenever a resume event is detected on USB. This is valid for both USB device and USB host modes. 0 = DISABLE 1 = ENABLE
1	RW	0x0	USB_WAKEUP_INT_ENB: USB wakeup interrupt enable. If this bit is enabled, interrupt is generated whenever USB wakeup event is generated. 0 = DISABLE 1 = ENABLE
0	RO	0x0	USB_WAKEUP_INT_STS: USB wakeup interrupt status. This bit is set whenever the USB wakes up from suspend (a wakeup event is generated). Software must write a 1 to clear this bit. Note that during the wakeup sequence, PHY clocks will be resumed from suspend. Software can check when the PHY clocks are resumed by reading the bit USB_PHY_CLK_VALID. There is also a separate interrupt generated when PHY clock is resumed if USB_PHY_CLK_VALID_INT_EN is set. During the wakeup sequence, first USB_WAKEUP_INT_STS will be set, and it will take some time for the PHY clock to resume, which can be detected by checking USB_PHY_CLK_VALID. 0 = UNSET 1 = SET

19.13.5.2 USB2_IF_USB_PHY_VBUS_SENSORS_0

This register controls the OTG VBUS sensors in the USB PHY. There are 4 VBUS sensors:

- A_VBUS_VLD
- A_SESS_VLD
- B_SESS_VLD
- B_SESS_END

The debounced status of each sensor can be read from the corresponding _STS bit field of the sensor in this register. The _CHG_DET field is set to 1 whenever a change is detected in the value of the _STS bit field of the corresponding sensor. If _INT_EN is set, then an interrupt is generated to the processor. This interrupt can be routed to CPU/AVP by appropriately writing the USBD bits in the interrupt controller registers.

In case software wants to override the value for a sensor, it can set the corresponding _SW_EN to 1, and set the corresponding sensor _SW_VALUE to 1 or 0 as per the requirement.

There are two debouncers for each sensor - DEBOUNCE_A and DEBOUNCE_B. The debounce values for them are controlled by the register UTMIP_DEBOUNCE_CFG0, fields UTMIP_BIAS_DEBOUNCE_A and UTMIP_BIAS_DEBOUNCE_B. For each sensor, we can select whether to use DEBOUNCE_A or DEBOUNCE_B by setting the field _DEB_SEL_B to the appropriate value (SEL_A or SEL_B).

Note: Do not set either UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.

USB PHY VBUS SENSORS Control Register

Offset: 0x404 | Read/Write: R/W | Reset: 0bx0000x00x0000x00x0000x00x0000x00

Bit	R/W	Reset	Description
30	RW	0x0	A_VBUS_VLD_WAKEUP_EN: A_VBUS_VLD wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on A_VBUS_VLD. 0 = DISABLE 1 = ENABLE
29	RW	0x0	A_VBUS_VLD_DEB_SEL_B: A_VBUS_VLD debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
28	RW	0x0	A_VBUS_VLD_SW_VALUE: A_VBUS_VLD software value. Software should write the appropriate value (1/0) to set/unset the A_VBUS_VLD status. This is only valid when A_VBUS_VLD_SW_EN is set. 0 = UNSET 1 = SET
27	RW	0x0	A_VBUS_VLD_SW_EN: A_VBUS_VLD software enable. Enable Software Controlled A_VBUS_VLD. Software sets this bit to drive the value in A_VBUS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
26	RO	X	A_VBUS_VLD_STS: A_VBUS_VLD status. This is set to 1 whenever A_VBUS_VLD sensor output is 1. 0 = UNSET 1 = SET
25	RO	0x0	A_VBUS_VLD_CHG_DET: A_VBUS_VLD change detect. This field is set by hardware whenever a change is detected in the value of A_VBUS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
24	RW	0x0	A_VBUS_VLD_INT_EN: A_VBUS_VLD interrupt enable. If this field is set to 1, an interrupt is generated whenever A_VBUS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
22	RW	0x0	A_SESS_VLD_WAKEUP_EN: A_SESS_VLD wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on A_SESS_VLD. 0 = DISABLE 1 = ENABLE
21	RW	0x0	A_SESS_VLD_DEB_SEL_B: A_SESS_VLD debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
20	RW	0x0	A_SESS_VLD_SW_VALUE: A_SESS_VLD software value. Software should write the appropriate value (1/0) to set/unset the A_SESS_VLD status. This is only valid when A_SESS_VLD_SW_EN is set. 0 = UNSET 1 = SET
19	RW	0x0	A_SESS_VLD_SW_EN: A_SESS_VLD software enable. Enable Software Controlled A_SESS_VLD. Software sets this bit to drive the value in A_SESS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
18	RO	X	A_SESS_VLD_STS: A_SESS_VLD status. This is set to 1 whenever A_SESS_VLD sensor output is 1. 0 = UNSET 1 = SET
17	RO	0x0	A_SESS_VLD_CHG_DET: A_SESS_VLD change detect. This field is set by hardware whenever a change is detected in the value of A_SESS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET
16	RW	0x0	A_SESS_VLD_INT_EN: A_SESS_VLD interrupt enable. If this field is set to 1, an interrupt is generated whenever A_SESS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
14	RW	0x0	B_SESS_VLD_WAKEUP_EN: B_SESS_VLD wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on B_SESS_VLD. 0 = DISABLE 1 = ENABLE
13	RW	0x0	B_SESS_VLD_DEB_SEL_B: B_SESS_VLD debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
12	RW	0x0	B_SESS_VLD_SW_VALUE: B_SESS_VLD software value. Software should write the appropriate value (1/0) to set/unset the B_SESS_VLD status. This is only valid when B_SESS_VLD_SW_EN is set. 0 = UNSET 1 = SET
11	RW	0x0	B_SESS_VLD_SW_EN: B_SESS_VLD software enable. Enable Software Controlled B_SESS_VLD. Software sets this bit to drive the value in B_SESS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
10	RO	X	B_SESS_VLD_STS: B_SESS_VLD status. This is set to 1 whenever B_SESS_VLD sensor output is 1. 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
9	RO	0x0	B_SESS_VLD_CHG_DET: B_SESS_VLD change detect. This field is set by hardware whenever a change is detected in the value of B_SESS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET
8	RW	0x0	B_SESS_VLD_INT_EN: B_SESS_VLD interrupt enable. If this field is set to 1, an interrupt is generated whenever B_SESS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
6	RW	0x0	B_SESS_END_WAKEUP_EN: B_SESS_END wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on B_SESS_END. 0 = DISABLE 1 = ENABLE
5	RW	0x0	B_SESS_END_DEB_SEL_B: B_SESS_END debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
4	RW	0x0	B_SESS_END_SW_VALUE: B_SESS_END software value. Software should write the appropriate value (1/0) to set/unset the B_SESS_END status. This is only valid when B_SESS_END_SW_EN is set. 0 = UNSET 1 = SET
3	RW	0x0	B_SESS_END_SW_EN: B_SESS_END software enable. Enable Software Controlled B_SESS_END Software sets this bit to drive the value in B_SESS_END_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
2	RO	X	B_SESS_END_STS: B_SESS_END status. This is set to 1 whenever B_SESS_END sensor output is 1. 0 = UNSET 1 = SET
1	RO	0x0	B_SESS_END_CHG_DET: B_SESS_END change detect. This field is set by hardware whenever a change is detected in the value of B_SESS_END. software writes a 1 to clear it 0 = UNSET 1 = SET
0	RW	0x0	B_SESS_END_INT_EN: B_SESS_END interrupt enable. If this field is set to 1, an interrupt is generated whenever B_SESS_END_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE

19.13.5.3 USB2_IF_USB_PHY_VBUS_WAKEUP_ID_0

This register controls the battery charger (VDCD_DET, VDAT_DET), VBUS_WAKEUP and ID sensors. The following sensors are in this register:

- VBUS_WAKEUP
- ID
- VDAT_DET
- VDCD_DET

The debounced status of each sensor can be read from the corresponding _STS bit field of the sensor in this register. The _CHG_DET field is set to 1 whenever a change is detected in the value of the _STS bit field of the corresponding sensor. If _INT_EN is set, then an interrupt is generated to the processor. This interrupt can be routed to CPU/AVP by appropriately writing the USBD bits in the interrupt controller registers.

In case software wants to override the value for a sensor, it can set the corresponding _SW_EN to 1, and set the corresponding sensor _SW_VALUE to 1 or 0 as per the requirement.

There are two debouncers for each sensor - DEBOUNCE_A and DEBOUNCE_B. The debounce values for them are controlled by the register UTMIP_DEBOUNCE_CFG0, fields UTMIP_BIAS_DEBOUNCE_A and UTMIP_BIAS_DEBOUNCE_B. For each sensor, we can select whether to use DEBOUNCE_A or DEBOUNCE_B by setting the field _DEB_SEL_B to the appropriate value (SEL_A or SEL_B).

Note: Do not set either UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.

There are debouncers for VDAT_DET and VDCD_DET. These use separate debouncers -CHRG_DEBOUNCE_PERIOD_A and CHRG_DEBOUNCE_PERIOD_B. The debounce values for them are controlled by the register UTMIP_CHRG_DEB_CFG0, fields UTMIP_CHRG_DEBOUNCE_PERIOD_A and UTMIP_CHRG_DEBOUNCE_PERIOD_B. For each sensor, we can select whether to use CHRG_DEBOUNCE_PERIOD_A or CHRG_DEBOUNCE_PERIOD_B by setting the field _DEB_SEL_B to the appropriate value (SEL_A or SEL_B).

Note: Do not set either UTMIP_CHRG_DEBOUNCE_PERIOD_A or UTMIP_CHRG_DEBOUNCE_PERIOD_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.

USB2 PHY VBUS Wakeup and ID Control Register

Offset: 0x408 | Read/Write: R/W | Reset: 0b00000x00xx000x00xx000x00x1000x00

Bit	R/W	Reset	Description
31	RW	0x0	DIV_DET_EN: Battery charger divider detection enable. This goes to the USB2OTG pad. 0 = DISABLE 1 = ENABLE
30	RW	0x0	VBUS_WAKEUP_WAKEUP_EN: VBUS_WAKEUP wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on VBUS_WAKEUP. 0 = DISABLE 1 = ENABLE
29	RW	0x0	VDCD_DET_DEB_SEL_B: VCDT_DET debounce A/B select. Selects the debounce value from UTMIP_CHRG_DEBOUNCE_PERIOD_A or UTMIP_CHRG_DEBOUNCE_PERIOD_B from the register UTMIP_CHRG_DEB_CFG0. 0 = SEL_A 1 = SEL_B
28	RW	0x0	VDCD_DET_SW_VALUE: VDCD_DET software value. Software should write the appropriate value (1/0) to set/unset the VDCD_DET status. This is only valid when VDCD_DET_SW_EN is set. 0 = UNSET 1 = SET
27	RW	0x0	VDCD_DET_SW_EN: VDCD_DET software enable. Enable Software Controlled VDCD_DET. Software sets this bit to drive the value in VDCD_DET_SW_VALUE to the USB controller 0 = DISABLE 1 = ENABLE
26	RO	X	VDCD_DET_STS: VDCD_DET status. This is set to 1 whenever VDCD_DET sensor output is 1. 0 = UNSET 1 = SET
25	RO	0x0	VDCD_DET_CHG_DET: VDCD_DET change detect. This field is set by hardware whenever a change is detected in the value of VDCD_DET. software writes a 1 to clear it 0 = UNSET 1 = SET
24	RW	0x0	VDCD_DET_INT_EN: VDCD_DET interrupt enable. If this field is set to 1, an interrupt is generated whenever VDCD_DET_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
23	RO	X	VOP_DIV2P7_DET:This read-only status bit from battery charging divider circuit of USB2OTG pad 0 = UNSET

Bit	R/W	Reset	Description
			1 = SET
22	RO	X	VOP_DIV2P0_DET:This read-only status bit from battery charging divider circuit of USB2OTG pad 0 = UNSET 1 = SET
21	RW	0x0	VDAT_DET_DEB_SEL_B: VDAT_DET debounce A/B select. Selects between the two debounce values UTMIP_CHRG_DEBOUNCE_PERIOD_A or UTMIP_CHRG_DEBOUNCE_PERIOD_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
20	RW	0x0	VDAT_DET_SW_VALUE: VDAT_DET software value. Software should write the appropriate value (1/0) to set/unset the VDAT_DET status. This is only valid when VDAT_DET_SW_EN is set. 0 = UNSET 1 = SET
19	RW	0x0	VDAT_DET_SW_EN: VDAT_DET software enable. Enable Software Controlled VDAT_DET. Software sets this bit to drive the value in VDAT_DET_SW_VALUE to the USB controller 0 = DISABLE 1 = ENABLE
18	RO	X	VDAT_DET_STS: VDAT_DET status. This is set to 1 whenever the VDAT_DET sensor output is 1. 0 = UNSET 1 = SET
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
15	RO	X	VON_DIV2P7_DET:This read-only status bit from battery charging divider circuit of USB2OTG pad 0 = UNSET 1 = SET
14	RO	X	VON_DIV2P0_DET:This read-only status bit from battery charging divider circuit of USB2OTG pad 0 = UNSET 1 = SET
13	RW	0x0	VBUS_WAKEUP_DEB_SEL_B: VBUS_WAKEUP debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
12	RW	0x0	VBUS_WAKEUP_SW_VALUE: VBUS wakeup software value. Software should write the appropriate value (1/0) to set/unset the VBUS_WAKEUP status. This is only valid when VBUS_WAKEUP_SW_EN is set. 0 = UNSET 1 = SET
11	RW	0x0	VBUS_WAKEUP_SW_EN: VBUS wakeup software enable. Enable Software Controlled VBUS_WAKEUP. Software sets this bit to drive the value in VBUS_WAKEUP_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
10	RO	X	VBUS_WAKEUP_STS: VBUS wakeup status. This is set to 1 whenever VBUS_WAKEUP sensor output is 1. 0 = UNSET 1 = SET
9	RO	0x0	VBUS_WAKEUP_CHG_DET: VBUS wakeup change detect. This field is set by hardware whenever a change is detected in the value of VBUS_WAKEUP. software writes a 1 to clear it 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
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

19.13.5.4 USB2_IF_USB_PHY_ALT_VBUS_STS_0

USB PHY Alternate VBUS Status Register

Offset: 0x40c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
8	X	VDCD_DET_ALT: VDCD_DET alternate status 0 = UNSET 1 = SET
7	X	VDAT_DET_ALT: VDAT_DET alternate status 0 = UNSET 1 = SET
6	X	A_SESS_VLD_ALT: A_SESS_VLD alternate status 0 = UNSET 1 = SET
5	X	B_SESS_VLD_ALT: B_SESS_VLD alternate status 0 = UNSET 1 = SET
4	X	ID_DIG_ALT: ID alternate status 0 = UNSET

Bit	Reset	Description
		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

19.13.5.5 USB2_IF_USB_ULPIS2S_CTRL_0

This register is used to set up parameters for ULPI null PHY mode.

Note: Current ULPI configuration registers use settings that support NV software drivers and should NOT be modified. If designing custom drivers, consult your NV representative prior to modifying these registers

ULPI NULL PHY Control Register

Offset: 0x418 | Read/Write: R/W | Reset: 0bxxxxxxxx0000xx0000000000xxxx0000

Bit	Reset	Description
23:20	0x0	ULPIS2S_CLAMP_LINE_DRIVE: The line drive value that should be sent into the line simulator during transmit clamping. The suggested value is 0x1: tri-state.
17	0x0	ULPIS2S_SLV1_CLAMP_XMIT: When set to 1, the outputs of the SLV1 transmit state machine are clamped to 0. This bit should be set before the SLV0 async reset is asserted to prevent meta-stability issues in the line simulator.
16	0x0	ULPIS2S_SLV0_CLAMP_XMIT: When set to 1, the outputs of the SLV0 transmit state machine are clamped to 0. This bit should be set before the SLV0 async reset is asserted to prevent meta-stability issues in the line simulator.
15	0x0	ULPIS2S_DISABLE_STP_PU: When set to 1 and in ULPIS2S mode, the pullup on the STP pin will NOT be active, even if the remote LINK asks to do so. In this case, an external pullup resistor would be required to ensure valid levels when the remote link is not powered.
14	0x0	ULPIS2S_SUPPORT_HS_KEEP_ALIVE: When enabled, the PHY will support HS KeepAlive packets. In that case, this would be the only thing that is supported in Opmode3. All other Opmode3 generate packets are not supported under any circumstances. 0 = DISABLE 1 = ENABLE
13	0x0	ULPIS2S_DISCON_DONT_CHECK_SE0: When enabled, the disconnect detection logic will only check that that the other side is 'driving' tri-state. It will not check whether or not the local side is driving SE0. 0 = DISABLE 1 = ENABLE
12	0x0	ULPIS2S_FORCE_ULPI_CLK_OUT: When enabled and ULPIS2S_ENA is ENABLED, the external ULPI_CLOCK pad will always carry the internal 60MHz clock, even if the interface is in shutdown mode. 0 = DISABLE 1 = ENABLE
11:8	0x0	ULPIS2S_SPARE: Reserved bits.
3	0x0	ULPIS2S_PLLU_MASTER_BLASTER60: When enabled, the PLLU 60MHz clock will be forced on. 0 = DISABLE 1 = ENABLE
2	0x0	ULPIS2S_SUPPORT_DISCONNECT: When disabled, the PHY will never detect a Disconnect.

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
1	0x0	ULPIS2S_SLV1_FORCE_DEVICE: When disabled, the slave port that is connected to the pins can be programmed to be host or a device depending on the value of the DpPulldown and DmPulldown bits in the OTG_CTRL ULPI register. When enabled, the values of those bits in the OTG_CTRL register is ignored and the port will always behave like a device. 0 = DISABLE 1 = ENABLE
0	0x0	ULPIS2S_ENA: When enabled, the ULPI link interface coming out of the USB2 controller enters a NULL PHY with two slaves. As a result the external pins will have a slave ULPI interface. When disabled, the ULPI link interface coming out of the USB2 controller go straight to the pins. 0 = DISABLE 1 = ENABLE

19.13.5.6 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: 0x41c | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	ULPIS2S_SLV1_VENDOR_ID: PHY vendor_id as seen by the external ULPI master
15:0	0x0	ULPIS2S_SLV1_PRODUCT_ID: PHY product_id as seen by the external ULPI master

19.13.5.7 USB2_IF_USB_INTER_PKT_DELAY_CTRL_0

Inter Packet Delay Control

This register controls the interpacket delay between two consecutive transmit packets in HS mode. This only applies to host mode as device never transmits two packets in a row.

Offset: 0x420 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0010010

Bit	Reset	Description
6:0	0x12	IP_DELAY_TX2TX_HS: HS Tx to Tx inter-packet delay. This is valid only for UHSIC PHY. Software should not change this.

19.13.5.8 USB2_IF_USB_RSM_DLY_0

USB Controller Resume Signalling Delay

Offset: 0x490 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0110100101111000

Bit	Reset	Description
15:0	0x6978	TIME_TO_RESUME: Send the resume back in number of 60 MHz cycles. Default gives 900 µs delay. Only applicable in host mode.

19.13.5.9 USB2_IF_SPARE_0

For ICUSB PADCTLS

Spare Register

Offset: 0x498 | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:16	0xffff	SPARE_HI: Spare register bits, defaulted to 1. SPARE_HI[0] - usb_port_reset_fix_en. Enable port-reset fix SPARE_HI[1] - hsic_conn_det_feature_enable. Enable HSIC connect detection
15:0	0x0	SPARE_LO: Spare register bits, defaulted to 0. SPARE_LO[4] - usb_hs_rsm_eop_fix_en. Enable HS_RESUME EOP fix SPARE_LO[5] - usb_port_suspend_fix_en

19.13.5.10 USB2_IF_ULPI_DIR_OVERRIDE_0

ULPI Override

Offset: 0x49c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1

Bit	Reset	Description
0	0x1	ULPI_DIR_OVERRIDE: By default, this bit is set. This will override ulpi_dir; i.e., when this bit is set, ulpi_dir is always asserted. Software needs to explicitly clear this bit, once slv1 Lp0 context is restored.

19.13.5.11 USB2_IF_USB2_NEW_CONTROL_0

USB Coherency and Memory Alignment Controls

Offset: 0x4c0 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx00001111xxxxxx00

Bit	Reset	Description
15:8	0xf	REQUEST_EXPIRY_COUNTER: Time to wait for coalescing the request.
1	0x0	MEM_ALIGNMENT_MUX_EN: Mux to select between Tegra 3 style (0) and Tegra K1 style (1) DMA request generation mechanism 0 = DISABLE 1 = ENABLE
0	0x0	COHERENCY_EN: Enable fence mechanism. 0 = DISABLE 1 = ENABLE

19.13.6 USB2 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

19.13.6.1 UTMIP REGISTER: PLL_CFG0

This register was used to configure PLL inside UTMIP block prior to Tegra K1 devices. This has been de-featured from UTMIP space and moved to CAR space. This register is used to configure the PHY PLL contained in the UTMIP module. Refer to the Clock and Reset Controller section for details on this register.

Note: This register has been moved to the Clock and Reset section of this document. PROGRAMMING THIS REGISTER HERE WILL HAVE NO EFFECT!

19.13.6.2 USB2_UTMIP_PLL_CFG1_0

UTMIP PLL and PLLU Configuration Register 1

Note: This register has been moved to the Clock and Reset section of this document. PROGRAMMING THIS REGISTER HERE WILL HAVE NO EFFECT!

This register was used to configure the PLL inside the UTMIP block prior to Tegra K1 devices. This register has been defeatured from UTMIP space and moved to CAR space. Refer to the Clock and Reset Controller section for details on this register.

19.13.6.3 USB2_UTMIP_XCVR_CFG0_0

UTMIP Transceiver Cell Configuration Register 0

Offset: 0x808 | Read/Write: R/W | Reset: 0b00100000001001010110010100000000

Bit	Reset	Description
31:25	0x10	UTMIP_XCVR_HSSLEW_MSB: Most significant bits of HS_SLEW.
24:22	0x0	UTMIP_XCVR_SETUP_MSB: Most significant bits of SETUP.
21	0x1	UTMIP_XCVR_LSBIAS_SEL: Low speed bias selection method for USB transceiver pad
20	0x0	UTMIP_XCVR_DISCON_METHOD: Disconnect method on the USB transceiver pad
19	0x0	UTMIP_FORCE_PDZI_POWERUP: Force PDZI input into power up.
18	0x1	UTMIP_FORCE_PDZI_POWERDOWN: Force PDZI input into power down. (Overrides FORCE_PDZI_POWERUP.)
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.

19.13.6.4 USB2_UTMIP_BIAS_CFG0_0

UTMIP Bias Cell Configuration Register 0

Offset: 0x80c | Read/Write: R/W | Reset: 0bx00000001100000000000110000000000

Bit	Reset	Description
30	0x0	UTMIP_IDDIG_C_VAL: See IDDIG_C_SEL.
29	0x0	UTMIP_IDDIG_C_SEL: 0: IdDig_c = IdDig_c.

Bit	Reset	Description
		1: IdDig_c = IDDIG_C_VAL.
28	0x0	UTMIP_IDDIG_B_VAL: See IDDIG_B_SEL.
27	0x0	UTMIP_IDDIG_B_SEL: 0: IdDig_b = IdDig_b. 1: IdDig_b = IDDIG_B_VAL.
26	0x0	UTMIP_IDDIG_A_VAL: See IDDIG_A_SEL.
25	0x0	UTMIP_IDDIG_A_SEL: 0: IdDig_a = IdDig_a. 1: IdDig_a = IDDIG_A_VAL.
24	0x0	UTMIP_HSDISCON_LEVEL_MSB: Most significant bit of UTMIP_HSDISCON_LEVEL, bit 2
23	0x1	UTMIP_IDPD_VAL: See IDPD_SEL.
22	0x1	UTMIP_IDPD_SEL: 0: Reserved. Refer to the PMC registers for this feature.
21	0x0	UTMIP_IDDIG_VAL: See IDDIG_SEL.
20	0x0	UTMIP_IDDIG_SEL: 0: IdDig = IdDig. 1: IdDig = IDDIG_VAL.
19	0x0	UTMIP_GPI_VAL: See GPI_SEL.
18	0x0	UTMIP_GPI_SEL: 0: StaticGpi = IdDig. 1: StaticGpi = GPI_VAL.
17:15	0x0	UTMIP_ACTIVE_TERM_OFFSET: Active termination control offset.
14:12	0x0	UTMIP_ACTIVE_PULLUP_OFFSET: Active 1.5K pullup control offset.
11	0x1	UTMIP_OTGPD: Power down OTG circuit.
10	0x1	UTMIP_BIASPD: Power down bias circuit.
9:8	0x0	UTMIP_VBUS_LEVEL_LEVEL: Vbus detector level.
7:6	0x0	UTMIP_SESS_LEVEL_LEVEL: SessionEnd detector level.
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.

19.13.6.5 USB2_UTMIP_HSRX_CFG0_0

UTMIP High Speed Receive Config 0

Offset: 0x810 | Read/Write: R/W | Reset: 0b10010001011001010011010000000000

Bit	Reset	Description
31:30	0x2	UTMIP_KEEP_PATT_ON_ACTIVE: Keep the stay alive pattern on active
29	0x0	UTMIP_ALLOW_CONSEC_UPDN: Allow consecutive ups and downs on the bits, debug only, set to 0.
28	0x1	UTMIP_REALIGN_ON_NEW_PKT: Realign the inertia counters on a new packet
27:24	0x1	UTMIP_PCOUNT_UPDN_DIV: The number of (edges-1) needed to move the sampling point
23:21	0x3	UTMIP_SQUELCH_EOP_DLY: Limit the delay of the squelch at EOP time
20	0x0	UTMIP_NO_STRIPPING: Do not strip incoming data
19:15	0xa	UTMIP_IDLE_WAIT: Number of idle cycles to declare IDLE.

Bit	Reset	Description
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

19.13.6.6 USB2_UTMIP_HSRX_CFG1_0

UTMIP High Speed Receive Config 1

Offset: 0x814 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxx010011

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

19.13.6.7 USB2_UTMIP_FSLSRX_CFG0_0

UTMIP Full and Low Speed Receive Config 0

Offset: 0x818 | Read/Write: R/W | Reset: 0b11111101010101001000010000101001

Bit	Reset	Description
31	0x1	UTMIP_FSLS_SE1_DRIBBLE_FILTER: One SE1, do not allow dribble
30	0x1	UTMIP_FSLS_SE1_FILTER: Filter SE1
29	0x1	UTMIP_FSLS_SERIAL_SE0_RCV
28:26	0x7	UTMIP_FSLS_UPR_DRIBBLE_SIZE: Do not allow <= dribble bits
25:23	0x2	UTMIP_FSLS_LWR_DRIBBLE_SIZE: Do not allow >= dribble bits
22	0x1	UTMIP_FSLS_EOP_ENDS_AT_SE0: Only look for transitioning out of EOP
21:16	0x14	UTMIP_FSLS_KCOUNT_MAX: Number of K bits in question
15	0x1	UTMIP_FSLS_KCOUNT_LIMIT: Limit the number of bit times a K can last
14	0x0	UTMIP_FSLS_ACTIVE_ON_FULL_SYNC: Require a full sync pattern to declare the data received
13:8	0x4	UTMIP_FSLS_IDLE_WAIT_MAX: 4 bits of SEO should exceed the time limit
7	0x0	UTMIP_FSLS_IDLE_WAIT_LIMIT: Enable the reset of the state machine on extended SE0
6:1	0x14	UTMIP_FSLS_IDLE_COUNT_MAX: 20 bits of idle should end the packet if FsLsIdleCountLimitCfg=1.
0	0x1	UTMIP_FSLS_IDLE_COUNT_LIMIT: Give up on packet if a long sequence of J

19.13.6.8 USB2_UTMIP_FSLSRX_CFG1_0

UTMIP Full and Low Speed Receive Config 1

Offset: 0x81c | Read/Write: R/W | Reset: 0bxxxxx01000100110011101000000000

Bit	Reset	Description
26	0x0	UTMIP_EARLY_LINE_STATE_FILTER: Assumes line state filtering table is inclusive, not exclusive
25:23	0x4	UTMIP_LS_BOUNCE_LENGTH: Number of clock cycle of LS stable
22:17	0x13	UTMIP_LS_EXTRACTION_COUNT: Phase count on which LS bits are extracted
16:11	0xe	UTMIP_LS_EOP_START_COUNT: Number of SEO clock cycles to block bit extraction
10:5	0x20	UTMIP_LS_SE0_COUNT: Only for this number of 60MHz of SEO and Idle to end packet
4	0x0	UTMIP_LS_LENIENT_DRIBBLE: Allow for large dribble in low speed mode
3	0x0	UTMIP_FS_LENIENT_DRIBBLE: Allow for large dribble in full speed mode
2	0x0	UTMIP_FS_WEAK_SYNC: Only look for a KK pattern, instead of KJKK
1	0x0	UTMIP_FS_DEBOUNCE: Whether full speed uses debouncing
0	0x0	UTMIP_FS_EOP_LENGTH: Whether full speed EOP is determined within 3 or 4 60MHz cycles 0: 3 60 MHz cycles 1: 4 60 MHz cycles

19.13.6.9 USB2_UTMIP_TX_CFG0_0

UTMIP Transmit Config Signals

Offset: 0x820 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxx00010000001000000000

Bit	Reset	Description
19	0x0	UTMIP_FS_PREAMBLE_J: Output enable sends an initial J before sync pattern
18	0x0	UTMIP_FS_POSTAMBLE_OUTPUT_ENABLE: Output enable turns off 1/2 cycle after
17	0x0	UTMIP_FS_PREAMBLE_OUTPUT_ENABLE: Output enable turns on 1/2 cycle before
16	0x1	UTMIP_FSLS_ALLOW_SOP_TX_STUFF_ERR: Allow SOP to be source of transmit error stuffing
15	0x0	UTMIP_HS_READY_WAIT_FOR_VALID
14:10	0x0	UTMIP_HS_TX_IPG_DLY
9	0x1	UTMIP_HS_DISCON_EOP_ONLY: Only check during EOP
8	0x0	UTMIP_HS_DISCON_DISABLE: Disable high speed disconnect
7	0x0	UTMIP_HS_POSTAMBLE_OUTPUT_ENABLE: Output enable turns off 1 cycle after
6	0x0	UTMIP_HS_PREAMBLE_OUTPUT_ENABLE: Output enable turns on 1 cycle before
5	0x0	UTMIP_SIE_RESUME_ON_LINESTATE: SIE, not macrocell, detects LineState change to resume
4	0x0	UTMIP_SOF_ON_NO_STUFF: SOF when OpMode 3 -- perhaps, when sending controller made packets
3	0x0	UTMIP_SOF_ON_NO_ENCODE: SOF when OpMode 2 -- not likely, for Chirp
2	0x0	UTMIP_NO_STUFFING: No bit stuffing, static programming
1	0x0	UTMIP_NO_ENCODING: No encoding, static programming
0	0x0	UTMIP_NO_SYNC_NO_EOP: Do not send SYNC or EOP

19.13.6.10 USB2_UTMIP_MISC_CFG0_0

UTMIP Miscellaneous Configurations

Offset: 0x824 | Read/Write: R/W | Reset: 0bx00000111110000000000000001111000

Bit	Reset	Description
30:27	0x0	UTMIP_DPDM_OBSERVE_SEL: Select DP/DM obs signals
26	0x0	UTMIP_DPDM_OBSERVE: Use DP/DM as obs bus
25	0x1	UTMIP_KEEP_XCVR_PD_ON_SOFT_DISCON
24	0x1	UTMIP_ALLOW_LS_ON_SOFT_DISCON
23	0x1	UTMIP_FORCE_FS_DISABLE_ON_DEV_CHIRP
22	0x1	UTMIP_SUSPEND_EXIT_ON_EDGE: Suspend exit requires edge or simply a value.
21	0x1	UTMIP_LS_TO_FS_SKIP_4MS: Do not block changes for 4 ms when going from LS to FS (should not happen)
20:19	0x0	UTMIP_INJECT_ERROR_TYPE: Force error insertion into RX path. (Used for IOBIST.) 0 = DISABLE 1 = BIT_ERR 2 = RX_ERR 3 = BIT_RX_ERR
18	0x0	UTMIP_FORCE_HS_CLOCK_ON: Force HS clock always on.
17	0x0	UTMIP_DISABLE_HS_TERM: Force HS termination inactive.
16	0x0	UTMIP_FORCE_HS_TERM: Force HS termination active.
15	0x0	UTMIP_DISABLE_PULLUP_DP: Force DP pullup inactive. (Overrides FORCE_PULLUP_DP.)
14	0x0	UTMIP_DISABLE_PULLUP_DM: Force DM pullup inactive. (Overrides FORCE_PULLUP_DM.)
13	0x0	UTMIP_DISABLE_PULLDN_DP: Force DP pulldown inactive. (Overrides FORCE_PULLDN_DP.)
12	0x0	UTMIP_DISABLE_PULLDN_DM: Force DM pulldown inactive. (Overrides FORCE_PULLDN_DM.)
11	0x0	UTMIP_FORCE_PULLUP_DP: Force DP pullup active.
10	0x0	UTMIP_FORCE_PULLUP_DM: Force DM pullup active.
9	0x0	UTMIP_FORCE_PULLDN_DP: Force DP pulldown active.
8	0x0	UTMIP_FORCE_PULLDN_DM: Force DM pulldown active.
7:5	0x3	UTMIP_STABLE_COUNT: Number of cycles of crystal clock of signal not changing to consider stable.
4	0x1	UTMIP_STABLE_ALL: Determines if all signal need to be stable to not change a config.
3	0x1	UTMIP_NO_FREE_ON_SUSPEND: Do not use free running terminations during suspend.
2	0x0	UTMIP_NEVER_FREE_RUNNING_TERMS: Ignore free-running terminations, even when no clock
1	0x0	UTMIP_ALWAYS_FREE_RUNNING_TERMS: Use free-running terminations at all time
0	0x0	UTMIP_COMB_TERMS: Use combinational terminations or synced through CLKXTAL

19.13.6.11 USB2_UTMIP_MISC_CFG1_0

UTMIP Miscellaneous Configurations

Offset: 0x828 | Read/Write: R/W | Reset: 0bx10000000001100110000000000100100

Bit	Reset	Description
30	0x1	UTMIP_PHY_XTAL_CLOCKEN: Selects whether to enable the crystal clock in the module.

Bit	Reset	Description
29	0x0	UTMIP_LINESTATE_BYPASS: Bypass LineState reclocking logic
28	0x0	UTMIP_LINESTATE_NEG: Use negative edge sync for line state
27	0x0	UTMIP_LINESTATE_XCVRSEL3: 0: Use FS filtering on line state when XcvrSel=3 1: Use LS filtering on line state when XcvrSel=3
26:25	0x0	UTMIP_OBS_SEL
24	0x0	UTMIP_FSLS_TDM
23	0x0	UTMIP_FORCE_IOBIST_CLK_ON
22:18	0x6	UTMIP_PLL_ACTIVE_DLY_COUNT: Reserved. Config moved to the Clock and Reset space.
17:6	0x600	UTMIP_PLLU_STABLE_COUNT: Reserved. Moved to the Clock and Reset space.
5	0x1	UTMIP_RX_ERROR_CNT_CLR
4	0x0	UTMIP_RX_ERROR_CNT_EN
3	0x0	UTMIP_FLIP_FSLS_POLARITY
2	0x1	UTMIP_SUSPEND_TERMSEL
1:0	0x0	UTMIP_XCVRSEL3: Bit 0: 0: 0xa5 -> treat as KeepAlive 1: treat as regular packet Bit 1: 0: Turn on FS EOP detection 1: Turn off FS EOP detection

19.13.6.12 USB2_UTMIP_DEBOUNCE_CFG0_0

UTMIP Avalid and Bvalid Debounce

Debounce values IdDig, Avalid, Bvalid, VbusValid, VbusWakeUp, and SessEnd. Each of these signals has its own debouncer. For each of those, one out of two debouncing times can be chosen (BIAS_DEBOUNCE_A or BIAS_DEBOUNCE_B.)

The values of DEBOUNCE_A and DEBOUNCE_B are calculated as follows:

0xffff -> No debouncing at all

ms = *1000 / (1/19.2MHz) / 4

So to program a 1 ms debounce for BIAS_DEBOUNCE_A:

BIAS_DEBOUNCE_A[15:0] = 1000 * 19.2 / 4 = 4800 = 0x12c0

Offset: 0x82c | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:16	0xffff	UTMIP_BIAS_DEBOUNCE_B: simulation value -- Used for interrupts
15:0	0xffff	UTMIP_BIAS_DEBOUNCE_A: simulation value -- Used for interrupts

19.13.6.13 USB2_UTMIP_BAT_CHRG_CFG0_0

UTMIP Battery Charger Configuration

Offset: 0x830 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxx000000xx000001

Bit	Reset	Description
13:8	0x0	UTMIP_CHRG_DEBOUNCE_TIMESCALE: Debouncer time scaling, factor-1 to slow down debouncing by. So 0 is 1, 1 is 2, etc.
5	0x0	UTMIP_OP_I_SRC_EN
4	0x0	UTMIP_ON_SRC_EN
3	0x0	UTMIP_OP_SRC_EN
2	0x0	UTMIP_ON_SINK_EN
1	0x0	UTMIP_OP_SINK_EN
0	0x1	UTMIP_PD_CHRG: Power-down charger circuit

19.13.6.14 USB2_UTMIP_SPARE_CFG0_0

UTMIP Spare Configuration Bits

Offset: 0x834 | Read/Write: R/W | Reset: 0b11111111111111111000000011111000

Bit	Reset	Description
31:0	-65288	UTMIP_SPARE: Spare register bits 0: HS_RX_IPG_ERROR_ENABLE 1: HS_RX_FLUSH_ALAP. Flush as late as possible 2: HS_RX_LATE_SQUELCH. Delay Squelch by 1 CLK480 cycle. 31 to 3: Reserved

19.13.6.15 USB2_UTMIP_XCVR_CFG1_0

UTMIP Transceiver Cell Configuration Register 1

Offset: 0x838 | Read/Write: R/W | Reset: 0bxxxx00000001100010000001000010101

Bit	Reset	Description
27:26	0x0	UTMIP_XCVR_RPU_RANGE_ADJ: 1.5k pull-up resistor range shift.
25:24	0x0	UTMIP_XCVR_HS_IREF_CAP: High-speed Iref cap control for bias current stability.
23:22	0x0	UTMIP_XCVR_SPARE: Spare bits for USB transceiver pad.
21:18	0x6	UTMIP_XCVR_TERM_RANGE_ADJ: Range adjustment on terminations.
17	0x0	UTMIP_RCTRL_SW_SET: Use a software override on RCTRL instead of automatic bias control.
16:12	0x8	UTMIP_RCTRL_SW_VAL: Encoded value to use on RCTRL when software override is enabled, 0 to 16 only.
11	0x0	UTMIP_TCTRL_SW_SET: Use a software override on TCTRL instead of automatic bias control
10:6	0x8	UTMIP_TCTRL_SW_VAL: Encoded value to use on TCTRL when software override is enabled, 0 to 16 only.
5	0x0	UTMIP_FORCE_PDDR_POWERUP: Force PDDR input into power up.
4	0x1	UTMIP_FORCE_PDDR_POWERDOWN: Force PDDR input into power down. (Overrides FORCE_PDDR_POWERUP.)
3	0x0	UTMIP_FORCE_PDCHRP_POWERUP: Force PDCHRP input into power up.
2	0x1	UTMIP_FORCE_PDCHRP_POWERDOWN: Force PDCHRP input into power down. (Overrides FORCE_PDCHRP_POWERUP.)
1	0x0	UTMIP_FORCE_PDDISC_POWERUP: Force PDDISC input into power up.
0	0x1	UTMIP_FORCE_PDDISC_POWERDOWN: Force PDDISC input into power down. (Overrides

Bit	Reset	Description
		FORCE_PDDISC_POWERUP.)

19.13.6.16 USB2_UTMIP_BIAS_CFG1_0

UTMIP Bias Cell Configuration Register 1

Offset: 0x83c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxx00000000101101

Bit	Reset	Description
13:8	0x0	UTMIP_BIAS_DEBOUNCE_TIMESCALE: Debouncer time scaling. Slows down debouncing by a factor-1. So 0 is 1, 1 is 2, etc.
7:3	0x5	UTMIP_BIAS_PDTRK_COUNT: Control the BIAS cell power down lag. The lag should be 20 µs. For a crystal clock of 13 MHz, it should be set to 5.
2	0x1	UTMIP_VBUS_WAKEUP_POWERDOWN: Reserved. See the PMC registers for this functionality.
1	0x0	UTMIP_FORCE_PDTRK_POWERUP: Force PDTRK input into power up.
0	0x1	UTMIP_FORCE_PDTRK_POWERDOWN: Force PDTRK input into power down. (Overrides FORCE_PDTRK_POWERUP.)

19.13.6.17 USB2_UTMIP_BIAS_STS0_0

UTMIP Bias Cell Status Register 0

Offset: 0x840 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	UTMIP_TCTRL: Thermal encoding output from USB bias pad.
15:0	X	UTMIP_RCTRL: Thermal encoding output from USB bias pad.

19.13.6.18 USB2_UTMIP_CHRG_DEB_CFG0_0

UTMIP VDcd_Det and VDat_Det Debounce

Debounce values VDcd_Det and VDat_Det. Each of these signals has its own debouncer. For each of those, 1 out of 2 debouncing times can be chosen (CHRG_DEBOUNCE_PERIOD_A or CHRG_DEBOUNCE_PERIOD_B).

The values of DEBOUNCE_PERIOD_A and DEBOUNCE_PERIOD_B are calculated as follows:

0xffff -> No debouncing at all

ms = *1000 / (1/19.2MHz) / 4

So to program a 1 ms debounce for CHG_DEBOUNCE_PERIOD:

CHG_DEBOUNCE_PERIOD[15:0] = 1000 * 19.2 / 4 = 4800 = 0x12c0

Offset: 0x844 | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:16	0xffff	UTMIP_CHRG_DEBOUNCE_PERIOD_B: simulation value -- Used for interrupts
15:0	0xffff	UTMIP_CHRG_DEBOUNCE_PERIOD_A: simulation value -- Used for interrupts

19.13.6.19 USB2_UTMIP_MISC_STS0_0

UTMIP MISC Status Register

Offset: 0x848 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
21	X	UTMIP_FS_DRV_EN: Indicates when the controller is driving on the bus
20:0	X	UTMIP_SPARE_FUSES: Spare fuses value to keep the connections preserved

19.13.6.20 USB2_UTMIP_PMC_WAKEUP0_0

UTMIP PMC Wakeup value

Offset: 0x84c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	R/W	Reset	Description
1	RO	X	UTMIP_LINE_WAKEUP_EVENT_INT_STS: Indicates the status of the PMC wakeup event
0	RW	0x0	UTMIP_LINE_WAKEUP_EVENT_INT_ENB: Interrupt mask for the PMC wakeup event

19.13.7 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

19.13.7.1 USB2_UHSIC_MISC_STS0_0

UHSIC SPARE Fuse Value

Offset: 0xc30 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
0	X	UHSIC_TX_HS_VLD: Indicates when the controller is driving on the bus

19.13.7.2 USB2_UHSIC_PMC_WAKEUP0_0

UHSIC PMC Wakeup Value

Offset: 0xc34 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	R/W	Reset	Description
1	RO	X	UHSIC_LINE_WAKEUP_EVENT_INT_STS: Indicates the status of the PMC wakeup event
0	RW	0x0	UHSIC_LINE_WAKEUP_EVENT_INT_ENB: Interrupt mask for the PMC wakeup event

19.13.7.3 USB2_QH_USB2D_QH_EP_n_OUT_0

USB2D Queue Head for OUT Endpoint n

There are 16 USB2D Queue Head for OUT Endpoint registers, where n = 0 through 15.

Offset: 0x1000 + (n * 0x80) | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint n. 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 n.

19.13.7.4 USB2_QH_USB2D_QH_EP_n_IN_0

There are 16 USB2D Queue Head for IN Endpoint registers, where n = 0 through 15.

USB2D Queue Head for IN Endpoint n

Offset: 0x1040 + (n * 0x80) | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint n. 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 n.

19.13.8 USB3 Controller Registers

19.13.8.1 USB2_CONTROLLER_2_USB2D_ID_0

USB2D Identification Register

Offset: 0x0 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	CIVERSION: Identifies the CI version
28:25	X	VERSION: Identifies the version of the core
24:21	X	REVISION: Revision number of the USB controller. This is set to 0x0.
20:16	X	TAG: Identifies the tag of the core
15:8	X	NID: One's complement version of ID. This field is set to 0xF9.
7:0	X	ID: Configuration number. This field is set to 0x06

19.13.8.2 USB2_CONTROLLER_2_USB2D_HW_HOST_0

USB2D Hardware Host Register

Offset: 0x8 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
3:1	X	NPORT: VUSB_HS_NUM_PORT-1: This host controller has only 1 port. So this field will always be 0.
0	X	HC: VUSB_HS_HOST: Indicates support for host mode. Set to 1.

19.13.8.3 USB2_CONTROLLER_2_USB2D_HW_DEVICE_0

USB2D Hardware Device Register

Offset: 0xc | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
5:1	X	DEVEP: VUSB_HS_DV_EP: Number of endpoints supported by this device controller. Set to 16. This includes control endpoint 0.

Bit	Reset	Description
0	X	DC: Device capable: Set to 1 indicating support for device mode.

19.13.8.4 USB2_CONTROLLER_2_USB2D_HW_TXBUF_0

USB2D Hardware TX Buffer Register

Offset: 0x10 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	TXCHANADD: VUSB_HS_TX_CHAN_ADD: Total number of address bits for the transmit buffer of each transmit endpoint. Set to 7. Each transmit buffer is 128 words deep.
15:8	X	TXADD: VUSB_HS_TX_ADD: Total number of address bits for the transmit buffer. Set to 11. The total depth of the transmit buffer is 2048 words.
7:0	X	TCBURST: VUSB_HS_TX_BURST: Maximum burst size supported by the transmit endpoints for data transfers. Set to 8.

19.13.8.5 USB2_CONTROLLER_2_USB2D_HW_RXBUF_0

USB2D RX Buffer HW Parameters Register

Offset: 0x14 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:8	X	RXADD: VUSB_HS_RX_ADD: Total number of address bits for the receive buffer. Set to 7. The total depth of the receive buffer is 128 words.
7:0	X	RXBURST: VUSB_HS_RX_BURST: Maximum burst size supported by the receive endpoints for data transfers. Set to 8.

19.13.8.6 USB2_CONTROLLER_2_USB2D_GPTIMER0LD_0

The host/device controller drivers can measure time-related activities using these timer registers. These registers are not part of the standard EHCI controller.

Offset: 0x80 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000)

Bit	Reset	Description
23:0	0x0	GPTIMER0LD: This field has the value to be loaded into the GPTCNT countdown timer on a reset action. The value in this register represents the time in microseconds minus 1 for the timer duration .

19.13.8.7 USB2_CONTROLLER_2_USB2D_GPTIMER0CTRL_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00XXXXXX (0b00xxxxx0xxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	0x0	GTPRUN: This bit enables the general-purpose timer to run. Setting or clearing this bit will not have an effect on the GPTCNT counter value.
30	WO	0x0	GPTRST: Writing a one to this bit reloads the GPTCNT with the value in GPTLD.
24	RW	0x0	GPTMODE: This bit selects between a single timer countdown and a looped countdown. In one-shot mode, the timer counts down to zero, generates an interrupt, and stops until the counter is reset by software. In repeat mode, the timer counts down to zero, generates an interrupt, and automatically reloads the counter to begin again.
23:0	RO	X	GPTCNT: This field has the value of the running timer.

19.13.8.8 USB2_CONTROLLER_2_USB2D_GPTIMER1LD_0

Offset: 0x88 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000)

Bit	Reset	Description
23:0	0x0	GPTIMER1LD: This field has the value to be loaded into the GPTCNT countdown timer on a reset action. The value in this register represents the time in microseconds minus 1 for the timer duration .

19.13.8.9 USB2_CONTROLLER_2_USB2D_GPTIMER1CTRL_0

Offset: 0x8c | Read/Write: R/W | Reset: 0x00XXXXXX (0b00xxxxx0xxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	0x0	GTPRUN: This bit enables the general-purpose timer to run. Setting or clearing this bit will not have an effect on the GPTCNT counter value.
30	WO	0x0	GPTRST: Writing a one to this bit reloads the GPTCNT with the value in GPTLD.
24	RW	0x0	GPTMODE: This bit selects between a single timer countdown and a looped countdown. In one-shot mode, the timer counts down to zero, generates an interrupt, and stops until the counter is reset by software. In repeat mode, the timer counts down to zero, generates an interrupt, and automatically reloads the counter to begin again.
23:0	RO	X	GPTCNT: This field has the value of the running timer.

19.13.8.10 USB2_CONTROLLER_2_USB2D_CAPLENGTH_0

USB2D Capability Register Length Register

Offset: 0x100 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
7:0	X	CAPLENGTH: Indicates which offset to add to the register base address at the beginning of the Operational Register. Set to 0x30.

19.13.8.11 USB2_CONTROLLER_2_USB2D_HCIVERSON_0

USB2D Host Interface Version Number Register

Offset: 0x102 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	HCIVERSION: Contains a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision. This host controller supports EHCI revision 1.00.

19.13.8.12 USB2_CONTROLLER_2_USB2D_HCSPARAMS_0

USB2D Host Control Structural Parameters Register

Offset: 0x104 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
27:24	X	N_TT: Number of Transaction Translators: Indicates the number of embedded transaction translators associated with the USB2.0 host controller. This field is always set to 1 indicating only 1 embedded TT is implemented in this implementation. This is a non-EHCI field to support embedded TT.
23:20	X	N_PTT: Number of Ports per Transaction Translator: Indicates the number of ports assigned to each

Bit	Reset	Description
		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.

19.13.8.13 USB2_CONTROLLER_2_USB2D_HCCPARAMS_0

USB2D Host Control Capability Parameters Register

Offset: 0x108 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
18	X	PPC: Per-Port Change Event Capability - RO. Default = 1b. This field indicates the support for per-port change events. This field is related to the USBCMD PPE field, USBSTS PPCI field, and USBINTR PPCE field.
17	X	LEN: Link Power Management Capability - RO. Default = 1b. This field indicates the support for LPM L1 state. This field is related to the USBCMD HIRD field, POSTSCx SSTS and DA fields, and HOSTPCx LEN, BA, and EPLPM fields.
15:8	X	EECP: EHCI Extended Capabilities Pointer: Indicates a capabilities list exists. A value of 00h indicates no extended capabilities are implemented. For this implementation this field is always "0".
7:4	X	IST: Isochronous Scheduling Threshold. This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit [7] is zero, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit [7] is a one, then host software assumes the host controller may cache an isochronous data structure for an entire frame. This field will always be "0".
2	X	ASP: Asynchronous Schedule Park Capability. 1 = (Default) the host controller supports the park feature for high-speed queue heads in the Asynchronous Schedule. The feature can be disabled or enabled and set to a specific level by using the Asynchronous Schedule Park Mode Enable and Asynchronous Schedule Park Mode Count fields in the USBCMD register. This field is always 1.
1	X	PFL: Programmable Frame List Flag. 0 = System software must use a frame list length of 1024 elements with this host controller. The USBCMD register Frame List Size field is a read-only register and must be set to zero. 1 = System software can specify and use a smaller frame list and configure the host controller via the USBCMD register Frame List Size field. The frame list must always be aligned on a 4K-page boundary. This requirement ensures that the frame list is always physically contiguous. This field will always be "1".

19.13.8.14 USB2_CONTROLLER_2_USB2D_DCIVERSION_0

USB2D Device Interface Version Number Register

Offset: 0x120 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	DCIVERSION: The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.

19.13.8.15 USB2_CONTROLLER_2_USB2D_DCCPARAMS_0

USB2D Device Control Capabilities Register

Offset: 0x124 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
8	X	HC: Host Capable: 1 = This controller is capable of operating as an EHCI compatible USB 2.0 host controller. This field is set to 1.
7	X	DC: Device Capable: 1 = Controller is capable of operating as USB 2.0 device. This field is set to 1.
5	X	LEN: Link Power Management Capability - RO. Default = 1b. This field indicates the support for LPM L1 state. This field is related to the DEVLCx ASUS, STL, BA, and NYT fields.
4:0	X	DEN: Device Endpoint Number: Number of endpoints built into the device controller. This is set to 16.

19.13.8.16 USB2_CONTROLLER_2_USB2D_EXTSTS_0

USB2D EXTSTS Register

Offset: 0x128 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000x

Bit	R/W	Reset	Description
4	RW	0x0	TI1: General Purpose Timer Interrupt 1 - RWC. Default = 0b. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero. Writing a one to this bit will clear it.
3	RW	0x0	TI0: General Purpose Timer Interrupt 0 - RWC. Default = 0b. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero. Writing a one to this bit will clear it.
2	RW	0x0	UPA: USB Host Periodic Interrupt (USBHSTPERINT) R/WC. This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set and the TD was from the periodic schedule. This bit is also set by the Host Controller when a short packet is detected AND the packet is on the periodic schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be zero. 0 = DISABLE 1 = ENABLE
1	RW	0x0	UAI: USB Host Asynchronous Interrupt (USBHSTASYNCINT) R/WC. This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set AND the TD was from the asynchronous schedule. This bit is also set by the Host when a short packet is detected AND the packet is on the asynchronous schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be zero. 0 = DISABLE 1 = ENABLE
0	RO	X	NAKI: NAK Interrupt Bit Read Only. This bit is read only. It is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and the corresponding TX/RX Endpoint NAK Enable bit are set. This bit is automatically cleared by hardware when the all the enabled TX/RX Endpoint NAK bits are cleared. 0 = DISABLE 1 = ENABLE

19.13.8.17 USB2_CONTROLLER_2_USB2D_USBEXTINTR_0

USB2D EXTINTR Register

Offset: 0x12c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000

Bit	Reset	Description
4	0x0	TIE1: General Purpose Timer Interrupt Enable 1 - RWC. Default = 0b. When this bit is a one, and the TI1 bit in the EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the TI1 bit.
3	0x0	TIE0: General Purpose Timer Interrupt Enable 0 - RWC. Default = 0b. When this bit is a one, and the TI1 bit in the EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the TI1 bit.
2	0x0	UPIE: UPIE Interrupt Enable. 1 = USB controller issues an interrupt if 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

19.13.8.18 USB2_CONTROLLER_2_USB2D_USBCMD_0

USB2D USB Command Register

Offset: 0x130 | Read/Write: R/W | Reset: 0bxxxx000000001000000x1x11x0000000

Bit	R/W	Reset	Description
27:24	RW	0x0	HIRD: Host Initiated Resume Duration RW. Default = 0000b. This has the same behavior as bits 7:4 of the BA field of the HOSTPCx register. When writing to this field all BA[7:4] fields of all HOSTPCx registers will be set to this value. This field is used by system software to specify the minimum amount of time the host controller will drive the K-state during a host-initiated resume from an LPM state (e.g., L1), and is conveyed to each LPM-enabled device (via the HIRD bits within an LPM Tokens bmAttributes field) upon entry into a low-power state. Note the host controller is required to drive resume signaling for at least the amount of time specified in the HIRD value conveyed to the device during any proceeding host-initiated resume. Also note that the host controller is not required to observe this requirement during device-initiated resumes. Encoding for this field is identical to the definition for the similarly named HIRD field within an LPM Token, specifically: a value 0000b equals 50 µs and each additional increment adds 75 µs. For example, the value 0001b equals 125 µs, and a value 1111b equals 1,175 µs (~1.2ms).
23:16	RW	0x8	ITC: Interrupt Threshold Control .Read/Write. Default 08h. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are shown below. Value Maximum Interrupt Interval 00h = Immediate (no threshold) 01h = 1 micro-frame 02h = 2 micro-frames 04h = 4 micro-frames 08h = 8 micro-frames 10h = 16 micro-frames 20h = 32 micro-frames 40h = 64 micro-frames 0 = IMMEDIATE 2 = ONE_MF 4 = TWO_MF 8 = EIGHT_MF 16 = SIXTEEN_MF 32 = THIRTY_TWO_MF 64 = SIXTY_FOUR_MF
15	RW	0x0	FS2: Bit 2 of Frame List Size.

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.
7	RO	X	LR: Light Host/Device Controller Reset (OPTIONAL). Not Implemented. This field will always be "0".
6	RW	0x0	IAA: Interrupt on Async Advance Doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is one, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to zero after it has set the Interrupt on Sync Advance status bit in the USBSTS register to one. Software should not write a one to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in Host mode. Writing a one to this bit when device mode is selected will have undefined results. 0 = CLEAR 1 = SET
5	RW	0x0	ASE: Asynchronous Schedule Enable. This bit controls whether the host controller skips processing the Asynchronous Schedule. 0 = Do not process the Asynchronous Schedule. 1 = Use the ASYNCLISTADDR register to access the Asynchronous Schedule. Only the host controller uses this bit. 0 = DISABLE 1 = ENABLE
4	RW	0x0	PSE: Periodic Schedule Enable. This bit controls whether the host controller skips processing the Periodic Schedule. 0 = Do not process the Periodic Schedule 1 = Use the PERIODICLISTBASE register to access the Periodic Schedule. Only the host controller uses this bit. 0 = DISABLE 1 = ENABLE
3:2	RW	0x0	FS1_FS0: Frame List Size 000 = Default. This field is Read/Write only if Programmable Frame List Flag in the HCCPARAMS registers is set to one. Hence this field is Read/Write for this implementation. This field specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. Note that this field is made up from USBCMD bits 15, 3, and 2. 000 = 1024 elements (4096 bytes) Default value 001 = 512 elements (2048 bytes) 010 = 256 elements (1024 bytes) 011 = 128 elements (512 bytes)

Bit	R/W	Reset	Description
			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

19.13.8.19 USB2_CONTROLLER_2_USB2D_USBSTS_0

USB2D USB Status Register

Offset: 0x134 | Read/Write: R/W | Reset: 0b00000000000000000x100x0000x0000

Bit	R/W	Reset	Description
31:16	RW	0x0	PPCI: Port-n Change Detect - RW. Default = 0000h. The definition for each bit is identical to the Port Change Detect field (bit 2 of this register) except these bits are specific to a given port, where bit 16 = Port 1, 17 = Port 2, etc. For example, if bit 17 is set to a one then a port change event was detected on Port 2. The N_PORTS field in HCSPARAMS specifies how many ports are exposed by the host controller and thus how many bits in this field are valid.
15	RW	0x0	AS: Asynchronous Schedule Status. This bit reports the current real status of the Asynchronous Schedule. When set to zero the asynchronous schedule status is disabled and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. If AS = ASE: 1= Enable Asynchronous Schedule 0= Disable Asynchronous Schedule Only used by the host controller. 0 = DISABLE 1 = ENABLE
14	RW	0x0	PS: Periodic Schedule Status. This bit reports the current real status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. If PS = PSE: 1 = Periodic Schedule is enabled

Bit	R/W	Reset	Description
			0 = Periodic Schedule is disabled Only used by the host controller. 0 = DISABLE 1 = ENABLE
13	RO	X	RCL: Reclamation. This is a read-only status bit used to detect an empty asynchronous schedule. Only used by the host controller. 0 = DISABLE 1 = ENABLE
12	RW	0x1	HCH: HCHalted. 1 = Default. This bit is a zero whenever the Run/Stop bit is a one. The Host Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g., internal error). Only used by the host controller. 0 = UNHALTED 1 = HALTED
11	RW	0x0	UALT_INT: ULPI alt_int Interrupt. 0 = Default. This interrupt bit is set when an RXCMD is received through the ULPI interface with bit 7 set (alt_int). The alt_int itself is set when an unmasked event occurs on any bit in the Carkit Interrupt Latch Register, in the ULPI PHY. The software should read the Carkit Interrupt Latch Register (Read to Clear) through the ULPI Viewport to check the source of the interrupt. Only present in designs where configuration constant VUSB_HS_PHY_ULPI = 1. 0 = NOT_ULPI_ALT_INT 1 = ULPI_ALT_INT
10	RW	0x0	ULPI_INT: ULPI Interrupt. This bit is set whenever an interrupt is received from ULPI PHY. Software writes 1 to clear it. 0 = NOT_ULPI_INT 1 = ULPI_INT
8	RW	0x0	SLI: DCSuspend. When a device controller enters a suspend state from an active state, this bit will be set to a 1. The device controller clears the bit upon exiting from a suspend state. Only used by the device controller. 0 = NOTSUSPEND 1 = SUSPENDED
7	RW	0x0	SRI: SOF Received. When the device controller detects a Start Of (micro) Frame, this bit will be set to a one. When an SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1ms in device FS mode and every 125 µs in HS mode and will be synchronized to the actual SOF that is received. Since device controller is initialized to FS before connect, this bit will be set at an interval of 1ms during the prelude to the connect and chirp. In host mode, this bit will be set every 125 µs and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it. This is a non-EHCI status bit. 0 = SOF_NOT_RCVD 1 = SOF_RCVD
6	RW	0x0	URI: USB Reset Received. When the device controller detects a USB Reset and enters the default state, this bit is set to a 1. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used by the device controller. 0 = NO_USB_RESET 1 = USB_RESET
5	RW	0x0	AAI: Interrupt and Asynchronous Advance. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the USBCMD register. This status bit indicates the assertion of that interrupt source. Only used by the host controller 0 = NOT_ADVANCED 1 = ADVANCED
4	RO	X	SEI: System Error. This bit is not used in this implementation and will always be set to "0". 0 = NO_ERROR 1 = ERROR
3	RW	0x0	FRI: Frame List Rollover. The Host Controller sets this bit to a 1 when the Frame List Index rolls over from its maximum value to 0. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the Frame List Size field of the USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [1 3] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a 1 every time FHINDEX

Bit	R/W	Reset	Description
			[12] toggles. Only used by the host controller. 0 = NO_ROLLOVER 1 = ROLLOVER
2	RW	0x0	PCI: Port Change Detect. The Host Controller sets this bit to a 1 when on any port a Connect Status occurs, a Port Enable/Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port. The Device Controller sets this bit to a one when the port controller enters the full or high-speed operational state. When the port controller exits the full or high-speed operational states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits respectively. This bit is not EHCI compatible. 0 = NO_PORT_CHANGE 1 = PORT_CHANGE
1	RW	0x0	UEI: USB Error Interrupt. This bit gets set by the Host/Device controller when completion of a USB transaction results in an error condition. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also ad its interrupt on complete (IOC) bit set. 0 = NO_ERROR 1 = ERROR
0	RW	0x0	UI: USB Interrupt. This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) as an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes. 0 = NO_INT 1 = INT

19.13.8.20 USB2_CONTROLLER_2_USB2D_USBINTR_0

USB2D USB Interrupt Enable Register

Offset: 0x138 | Read/Write: R/W | Reset: 0b0000000000000000xxxx00x000000000

Bit	Reset	Description
31:16	0x0	PPCE: Port-n Change Detect Enable - RW. Default = 0000h. The definition for each bit in this field is identical to bit 2 of this register (Port Change Interrupt Enable) except these bits are specific to a given port, where bit 16 = Port 1, 17 = Port 2, etc. For example, if bit 17 is set (1b) then a port change event was detected on Port 2. When a bit in this field is a one, and the corresponding Port-n Change Detect bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Port-n Change Detect bit.
11	0x0	UALTIE: ULPI alt_int Interrupt Enable. 1 = USB controller issues an interrupt if ULPI_ALT_INT bit in USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the ULPI_ALT_INT bit. 0 = DISABLE 1 = ENABLE
10	0x0	ULPIE: ULPI Interrupt Enable. 1 = USB controller issues an interrupt if ULPI_INT bit in USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the ULPI_INT bit. 0 = DISABLE 1 = ENABLE
8	0x0	SLE: Sleep Enable. 1 = Device controller issues an interrupt if DCSuspend bit in USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the DCSuspend bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE
7	0x0	SRE: SOF Received Enable. 1 = Device controller issues an interrupt if SOF Received bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the SOF Received bit. 0 = DISABLE 1 = ENABLE
6	0x0	URE: USB Reset Enable.1 = Device controller issues an interrupt if USB Reset Received bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the USB Reset Received bit. Only used by the device controller.

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
5	0x0	AAE: Interrupt on Asynchronous Advance Enable. 1 = the host controller issues an interrupt at the next interrupt threshold if Interrupt on Async Advance bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the Interrupt on Async Advance bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
4	0x0	SEE: System Error Enable. 1 = Host/device controller issues an interrupt if the System Error bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the System Error bit. 0 = DISABLE 1 = ENABLE
3	0x0	FRE: Frame List Rollover Enable. 1 = Host controller issues an interrupt if Frame List Rollover bit in the USBSTS register = 1. The interrupt is acknowledged by software clearing the Frame List Rollover bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
2	0x0	PCE: Port Change Detect Enable. 1 = Host/device controller issues an interrupt if Port Change Detect bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the Port Change Detect bit. 0 = DISABLE 1 = ENABLE
1	0x0	UEE: USB Error Interrupt Enable. 1 = Host controller issues an interrupt at the next interrupt threshold if the USBERRINT bit in USBSTS = 1. The interrupt is acknowledged by software clearing the USBERRINT bit in the USBSTS register. 0 = DISABLE 1 = ENABLE
0	0x0	UE: USB Interrupt Enable. 1 = Host/device issues an interrupt at the next interrupt threshold if the USBINT bit in USBSTS = 1. The interrupt is acknowledged by software clearing the USBINT bit. 0 = DISABLE 1 = ENABLE

19.13.8.21 USB2_CONTROLLER_2_USB2D_FRINDEX_0

USB2D USB Frame Index Register

Offset: 0x13c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
13:0	X	FRINDEX: Frame Index. The value in this register increments at the end of each time frame (micro-frame). Bits [N: 3] are used for the Frame List current index. Each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in host mode. USBCMD [Frame List Size] Number Elements N 000b (1024) 12 001b (512) 11 010b (256) 10 011b (128) 9 100b (64) 8 101b (32) 7 110b (16) 6 111b (8) 5 In device mode, the value is the current frame number of the last frame transmitted. It is not used as an index. In either mode, bits 2:0 indicate the current micro-frame.

19.13.8.22 USB2_CONTROLLER_2_USB2D_PERIODICLISTBASE_0

USB2D Host Controller Frame List Base Address Register

Offset: 0x144 | Read/Write: R/W | Reset: 0b00000000000000000000xxxxxxxxxxxx

Bit	Reset	Description
31:25	0x0	USBADR: Device mode. The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS request.
24	0x0	USBADRA: Device Address Advance. Default=0. When this bit is 0, any writes to USBADR are instantaneous. When this bit is written to a 1 at the same time or before USBADR is written, the write to the USBADR field is staged and held in a hidden register. After an IN occurs on endpoint 0 and is ACKed, USBADR will be loaded from the holding register. Hardware will automatically clear this bit on the following conditions: 1) IN is ACKed to endpoint 0. (USBADR is updated from staging register). 2) OUT/SETUP occur to endpoint 0. (USBADR is not updated). 3) Device Reset occurs (USBADR is reset to 0). Note: After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field. This mechanism will ensure this specification is met when the DCD cannot write of the device address within 2 ms from the SET_ADDRESS status phase. If the DCD writes the USBADR with USBADRA=1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR will be programmed instantly at the correct time and meet the 2ms USB requirement.
31:12	0x0	BASEADR: Host mode: This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. HCD loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register are combined with the Frame Index Register (FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence. Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.

19.13.8.23 USB2_CONTROLLER_2_USB2D_ASYNCLISTADDR_0

USB2D Next Asynchronous List Address Register

Offset: 0x148 | Read/Write: R/W | Reset: 0b00000000000000000000000000xxxxx

Bit	Reset	Description
31:11	0x0	EPBASE: Device mode. This register contains the address of the top of the endpoint list in system memory. These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Heads (QH). Only used by the device controller.
31:5	0x0	ASYBASE: Host mode. This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (OH). Only used by the host controller.

19.13.8.24 USB2_CONTROLLER_2_USB2D_ASYNCTTSTS_0

USB2D Asynchronous Buffer Status for Embedded TT Register

Offset: 0x14c | Read/Write: R/W | Reset: 0bx0000000xxxxxxxxxxxxxxxxxxxxxx0x

Bit	R/W	Reset	Description
30:24	RW	0x0	TTHA: Internal TT Hub Address representation. This field is used to match the Hub Address field in QH and siTD to determine if the packet is routed to the internal TT for directly attached FS/LS devices.If the Hub Address in the QH or siTD does not match this address then the packet will be broadcast on the High Speed ports destined a downstream High Speed hub with the address in QH/siTD.
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

Bit	R/W	Reset	Description
			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.

19.13.8.25 USB2_CONTROLLER_2_USB2D_BURSTSIZE_0

USB2D Burst Size Register

Offset: 0x150 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000100000001000

Bit	Reset	Description
15:8	0x8	TXPBURST: Programmable TX Burst Length. (Read/Write). This register represents the maximum length of a burst in 32-bit words while moving data from system memory to the USB bus.
7:0	0x8	RXPBURST: Programmable RX Burst Length. (Read/Write). This register represents the maximum length of a burst in 32-bit words while moving data from the USB bus to system memory.

19.13.8.26 USB2_CONTROLLER_2_USB2D_TXFILLTUNING_0

USB2D Transmit Fill Tuning Register

Offset: 0x154 | Read/Write: R/W | Reset: 0bxxxxxxxxxx000010xxx0000000000000

Bit	Reset	Description
21:16	0x2	TXFIFOTHRES: FIFO Burst Threshold. (Read/Write). This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USBMODE register is set.
12:8	0x0	TXSCHHEALTH: Scheduler Health Counter. (Read/Write To Clear) [Default = 0]. This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max out at 31.
7:0	0x0	TXSCHOH: Scheduler Overhead. (Read/Write) [Default = 0]. This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267 µs when a device is connected in High-Speed Mode. The time unit represented in this register is 6.333 µs when a device is connected in Low/Full Speed Mode.

19.13.8.27 USB2_CONTROLLER_2_USB2D_ICUSB_CTRL_0

USB2D ICUSB Control Register

This register enables and controls the ICUSB FS/LS transceiver.

Offset: 0x15c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000

Bit	Reset	Description
3	0x0	IC_ENB1: ICUSB transceiver enable. This bit enables the ICUSB transceiver. To enable the interface, the bits PTS must be set to 11 in the PORTSCx. Writing a '1' to this bit selects the IC_USB interface. 0 = DISABLE 1 = ENABLE
2:0	0x0	IC_VDD1: ICUSB voltage select. It selects which voltage is being supplied to the ICUSB peripheral. 000 = No voltage 001 = 1.0V – reserved 010 = 1.2V - reserved 011 = 1.5V - reserved 100 = 1.8V 101 = 3.0V 110 = reserved 111 = reserved The Voltage negotiation should happen between enabling port power (PP) and asserting the run/stop bit in register.

19.13.8.28 USB2_CONTROLLER_2_USB2D_ULPI_VIEWPORT_0

USB2D ULPI Viewport Register

This register provides indirect access to the ULPI PHY register set. Although the USB controller performs access to the ULPI PHY register set, there may be extraordinary circumstances where software may need direct access.

Note: WRITES TO THE ULPI THROUGH THE VIEWPORT CAN SUBSTANTIALLY HARM STANDARD USB OPERATIONS. CURRENTLY NO USAGE MODEL HAS BEEN DEFINED WHERE SOFTWARE SHOULD NEED TO EXECUTE WRITES DIRECTLY TO THE ULPI PHY. SEE EXCEPTION REGARDING OPTIONAL FEATURES BELOW.

Note: EXECUTING READ OPERATIONS THROUGH THE ULPI VIEWPORT SHOULD HAVE NO HARMFUL SIDE EFFECTS TO STANDARD USB OPERATIONS.

There are two operations that can be performed with the ULPI Viewport: wakeup and read /write operations.

The wakeup operation is used to put the ULPI interface into normal operation mode and re-enable the clock, if necessary. A wakeup operation is required before accessing the registers when the ULPI interface is operating in low power mode, serial mode, or Carkit mode.

The ULPI state can be determined by reading the sync state bit (ULPI_SYNC_STATE). If this bit is a one, then ULPI interface is running in normal operation mode and can accept read/write operations. If the ULPI_SYNC_STATE indicates a 0 then then read/write operations will not be able to execute. Undefined behavior will result if ULPI_SYNC_STATE = 0 and a read or write operation is performed.

To execute a wakeup operation, write all 32 bits of the ULPI Viewport where ULPI_PORT is constructed appropriately and the ULPI_WAKEUP bit is a 1 and ULPI_RUN bit is a 0. Poll the ULPI Viewport until ULPI_WAKEUP is zero for the operation to complete.

To execute a read or write operation, write all 32 bits of the ULPI Viewport where ULPI_DATA_WR, ULPI_REG_ADDR, ULPI_PORT, ULPI_RD_WR are constructed appropriately and the ULPI_RUN bit is a 1. Poll the ULPI Viewport until ULPI_RUN is zero for the operation to complete. Once ULPI_RUN is zero, the ULPI_DATA_RD will be valid if the operation was a read.

The polling method above can be changed to interrupt driven using the ULPI interrupt defined in the USBSTS and USBINTR registers. When a wakeup or read/write operation is complete, the ULPI_INT interrupt will be set.

There are several optional features that may need to be enabled or disabled by system software as part of system configuration. These bits are contained in the Interface and OTG Control registers of the ULPI PHY register set. These registers also contain bits which are controlled by the link dynamically and therefore should be only modified by system software using the Set/Clear access method. Direct writes to these registers could have harmful side effects to the standard USB operations. The optional bits are as follows: Bits 3 through 7 in the Interface Control register and Bits 6 and 7 in the OTG Control register.

Refer to the ULPI Specification Revision 1.1 for further information on the use of the optional features.

Offset: 0x160 | Read/Write: R/W | Reset: 0b000xx00000000000xxxxxxxx00000000

Bit	R/W	Reset	Description
31	RW	0x0	ULPI_WAKEUP: ULPI Wakeup. Writing the 1 to this bit will begin the wakeup operation. The bit will automatically transition to 0 after the wakeup is complete. Once this bit is set, the driver cannot set it back to 0. Note: The driver must never execute a wakeup and a read/write operation at the same time. 0 = CLEAR 1 = SET
30	RW	0x0	ULPI_RUN: ULPI read/write Run. Writing the 1 to this bit will begin the read/write operation. The bit will automatically transition to 0 after the read/write is complete. Once this bit is set, the driver cannot set it back to 0. Note: The driver must never execute a wakeup and a read/write operation at the same time. 0 = CLEAR 1 = SET
29	RW	0x0	ULPI_RD_WR: ULPI read/write control. (0) Read; (1) Write. This bit selects between running a read or write operation. 0 = READ 1 = WRITE
27	RO	X	ULPI_SYNC_STATE: ULPI sync state. (1) Normal Sync. State. (0) In another state (i.e., Carkit, serial, low power). This bit represents the state of the ULPI interface. 0 = NOT_NORMAL 1 = NORMAL
26:24	RW	0x0	ULPI_PORT: ULPI PHY port number. This field should be always written as 0.
23:16	RW	0x0	ULPI_REG_ADDR: ULPI PHY register address. When doing a read or write operation to the ULPI PHY, the address of the ULPI PHY register being accessed is written to this field.
15:8	RO	X	ULPI_DATA_RD: ULPI PHY data read. The data from the ULPI PHY register can be read from here after the read operation completes.
7:0	RW	0x0	ULPI_DATA_WR: ULPI PHY data write. The data to write to the ULPI PHY register is written here.

19.13.8.29 USB2_CONTROLLER_2_USB2D_PORTSC1_0

USB2D Port Status/Control 1 Register

Offset: 0x174 | Read/Write: R/W | Reset: 0b0000000xx0000000xxx1xx00x0xx010x

Bit	R/W	Reset	Description
31:25	RW	0x0	DA: Device Address. Default = 0000000b. The 7-bit USB device address for the device attached to an immediately downstream of the associated root port. A value of zero indicates no device is present or software support for this feature is not present. This is used by the Controller when sending the LPM token. This field is only valid when the core is operating in host mode. If in device mode, it will be read only and always equal to 0000000b.

Bit	R/W	Reset	Description
24:23	RO	X	SSTS: Suspend Status. Default = 00b. These two bits are used by software to determine whether an L1-based suspend request was successful, specifically: 00b - L1 state entered with success. ACK received from peripheral. 01b - NYET received from peripheral. It was not able to enter L1 state this time. 10b - L1 state not supported by peripheral. STALL received. 11b - Peripheral did not respond or an error occurred. The value of this field is only valid when the port resides in the L0 state - that is, the meaning of these bits is invalid whenever bit 7 of this register (SUSP) is one. Ideally, the Controller driver should read this register if it receives an interrupt after issuing a suspend using L1 support. In case of a non-success a port change interrupt will be fired and this field should be checked for a possible L1 failure. This field is only valid when the core is operating in host mode. If in device mode it will be always equal to 00b. 0 = L1STATE_ENTERED 1 = NYET_PERIPH 2 = L1STATE_NOT_SUPPORTED 3 = PERIPH_NORESP_ERR
22	RW	0x0	WKOC: Default = 0b. Wake on Over-current Enable: Writing this bit to a one enables the port to be sensitive to over-current conditions as wake-up events. This field is zero if Port Power (PP) is zero. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined behavior. 0 = DISABLE 1 = ENABLE
21	RW	0x0	WKDS: Wake on Disconnect Enable: Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This field is zero if Port Power (PP) is zero or in device mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined behavior. This bit should not be written to 1 if there is no device connected. After the device disconnect is detected, this bit should be cleared to 0. 0 = DISABLE 1 = ENABLE
20	RW	0x0	WKCN: Wake on Connect Enable: Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This field is zero if Port Power (PP) is zero or in device mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined behavior. This bit should not be written to 1 while the device is connected. After the device connection is detected, this bit should be cleared to 0. 0 = DISABLE 1 = ENABLE
19:16	RW	0x0	PTC: Port Test Control: Any value other than zero indicates that the port is operating in test mode. Value Specific Test. 0000b: Not enabled. 0001b: J_ STATE. 0010b: K_STATE. 0011b: SEQ_NAK. 0100b: Packet. 0101b: FORCE_ENABLE. 0110b to 1111b: Reserved. Refer to Chapter 7 of the USB Specification Revision 2.0 for details on each test mode. 0 = NORMAL_OP 1 = TEST_J 2 = TEST_K 3 = TEST_SE0_NAK 4 = TEST_PKT 5 = TEST_FORCE_ENABLE
15:14	RO	X	PIC: Port Indicator Control: This field is not supported in the current implementation. Please use a GPIO if you wish to use Port Indicators.
13	RO	X	PO: Port Owner. Port owner handoff is not implemented in this design, therefore this bit will always be 0.

Bit	R/W	Reset	Description
12	RW	0x1	PP: Port Power: The function of this bit depends on the value of the Port Power Switching (PPC) field in the HCSPARAMS register. The behavior is as follows: PPC PP Operation 0b 0b Read Only. A device controller with no OTG capability does not have port power control switches. 1b 1b/0b RW. Host/OTG controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (i.e., PP equals a 0), the port is non-functional and will not report attaches, detaches, etc. When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitioned by the host controller driver from a one to a zero (removing power from the port). 0 = NOT_POWERED 1 = POWERED
11:10	RO	X	LS: These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. The encoding of the bits is: 00b = SE0 01b = K-state 10b = J-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 = K_STATE 2 = J_STATE 3 = UNDEFINED
9	RW	0x0	SLP: Suspend using L1 - RW. Default = 0b. When this bit is set to '1' and a non-zero Device Address (DA) is specified Controller will instigate L1 entry during suspend (bit 7) and L1 exit during resume (bit 6). When set to zero the Controller will use the legacy (L2) mechanism. Software should only set this bit when the device attached immediately downstream supports L1 transitions. When acting as device, this bit is read-only and set to '1' by the hardware when the Controller enters is L1 state (LPM token received and accepted). Note: HSP is redundant with PSPD(27:26). This bit is not defined in the EHCI specification.
8	RW	0x0	PR: This field is zero if Port Power (PP) is zero. In Host Mode: Read/Write. 1=Port is in Reset. 0=Port is not in Reset. When software writes a one to this bit the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver. In Device Mode: This bit is a read only status bit. Device reset from the USB bus is also indicated in the USBSTS register. 0 = NOT_USB_RESET 1 = USB_RESET

Bit	R/W	Reset	Description
7	RO	X	SUSP: Port suspend. 1=Port in suspend state. 0=Port not in suspend state. In Host Mode: Read/Write. Port Enabled Bit and Suspend bit of this register define the port states as follows: Bits [Port Enabled, Suspend] Port State 0x Disable 10 Enable 11 Suspend When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to zero when software sets the Force Port Resume bit to zero. A write of zero to this bit is ignored by the host controller. If host software sets this bit to a one when the port is not enabled (i.e., Port enabled bit is a zero) the results are undefined. This field is zero if Port Power (PP) is zero in host mode. In Device Mode: Read Only. This bit is a read only status bit. 0 = NOT_SUSPEND 1 = SUSPEND
6	RW	0x0	FPR: Force Port Resume. 1= Resume detected/driven on port. 0=No resume (K state) detected/driven on port. In Host Mode: Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver. Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. This bit remains a one until the port has switched to the high-speed idle. Writing a zero has no effect because the port controller will time the resume operation to clear the bit when the port control state switches to HS or FS idle. This field is zero if Port Power (PP) is zero in host mode. This bit is not-EHCI compatible. In Device mode: After the device has been in Suspend State for 5ms or more, software must set this bit to one to drive resume signaling before clearing. The Device Controller will set this bit to one if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. Software should ensure that the PHY clock is operational before writing a 1 to this bit to start the resume sequence. This is true for both Device and Host modes. 0 = NO_RESUME 1 = RESUME
5	RO	X	OCC: Over-current Change: Not supported 0 = NO_CHANGE 1 = CHANGE
4	RO	X	OCA: Over-current Active: Not supported 0 = NO_OVER_CURRENT 1 = OVER_CURRENT
3	RW	0x0	PEC: Port Enable/Disable Change: 1=Port enabled/disabled status has changed. 0=No change. In Host Mode: For the root hub, this bit gets set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a one to it. This field is zero if Port Power (PP) is zero. In Device mode: The device port is always enabled. (This bit will be zero) 0 = NO_CHANGE 1 = CHANGE
2	RW	0x1	PE: Port Enabled/Disabled: 1=Enable. 0=Disable (default) In Host Mode: Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0b)

Bit	R/W	Reset	Description
			downstream propagation of data is blocked except for reset. This field is zero if Port Power (PP) is zero in host mode. In Device Mode: The device port is always enabled. (This bit will be one) 0 = PORT_DISABLED 1 = PORT_ENABLED
1	RW	0x0	CSC: Connect Status Change: 1 =Change in Current Connect Status. 0=No change (default). In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (i.e., the bit will remain set). Software clears this bit by writing a one to it. This field is zero if Port Power (PP) is zero in host mode. This bit is undefined in device controller mode. 0 = NO_CHANGE 1 = CHANGE
0	RO	X	CCS: Current Connect Status: In Host Mode: 1=Device is present on port. 0=No device is present (default). This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set. This field is zero if Port Power (PP) is zero in host mode. In Device Mode: 1=Attached 0=Not Attached (default) A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended. 0 = NOT_CONNECTED 1 = CONNECTED

19.13.8.30 USB2_CONTROLLER_2_USB2D_HOSTPC1_DEVLC_0

USB2D Device Mode LPM Behavior and Control Register

This register controls the LPM behavior and the behavior of each USB port in device mode only. This register shares the same address with the HOSTPC1 register (which is used only in host mode).

USB2D Host Mode LPM Behavior and Control Register

This register controls the LPM behavior and controls the behavior of each USB port in host mode only. This register shares the same address with the DEVLC register (which is used only in device mode).

Offset: 0x1b4 | Read/Write: R/W | Reset: 0b0000xxx0000000000000xxxxxxxxxxx0

Bit	R/W	Reset	Description
31:29	RW	0x0	PTS: Parallel transceiver select. This bit is not defined in the EHCI specification. 0 = UTMI 1 = RESERVED 2 = ULPI 3 = ICUSB_SER 4 = HSIC
28	RW	0x0	STS: Serial transceiver not selected. This is the only value supported. This bit is not defined in the EHCI specification. 0 = PARALLEL_IF 1 = SERIAL_IF
27	RO	X	PTW: Parallel Transceiver Width. Fixed to 0. This bit is not defined in the EHCI specification. 0 = EIGHT_BIT 1 = RESERVED

Bit	R/W	Reset	Description
26:25	RO	X	PSPD: This register field indicates the speed at which the port is operating. 00 = Full Speed 01 = Low Speed 10 = High Speed. This bit is not defined in the EHCI specification. 0 = FULL_SPEED 1 = LOW_SPEED 2 = HIGH_SPEED 3 = RESERVED
24	RW	0x0	ALPD: Auto Low Power While Disconnect - RW. Default = 0b. If set, this feature will be enabled and every time the port enters the disconnect state it will also enter in low power state, disabling the transceiver clock. The behavior will be same as if the PHCD bit was enabled(in fact this bit will be set to 1 as soon as low power mode is enabled).When this field is set the WKCN field of PORTSCx register(Wake on Connect Enable) will also be set. This way the core will wake up in case a connect is detected. There will be a delay between the detection of a disconnect and actually enter in low power mode. This delay can be controlled by writing to the ALPDD field in the USBMODE register. 0 = DONT_AUTO_LOW_POWER_WHILE_DISCONNECT 1 = AUTO_LOW_POWER_WHILE_DISCONNECT
23	RW	0x0	PFSC: Port Force Full Speed Connect - RW. Default = 0b. Writing this bit to a '1b' will force the port to only connect at Full Speed. It disables the chirp sequence that allows the port to identify itself as High Speed. This is useful for testing FS configurations with an HS host, hub or device. This bit is not defined in the EHCI specification. 0 = DONT_FORCE_FULL_SPEED 1 = FORCE_FULL_SPEED
22	RW	0x0	PHCD: PHY Low Power Suspend - Clock disable: Writing this bit to a 1 will disable the PHY clock. Write a 0 enables it. Reading this bit will indicate the status of the PHY clock. NOTE: The PHY clock cannot be disabled if it is being used as the system clock. In device mode, the PHY can be put into Low Power Clock Disable when the device is not running (USBCMD RS=0b) or the host has signaled suspend (PORTSCx SUSP=1b). Low Power Clock Disable will be cleared automatically when the host has signaled resume. Before forcing a resume from the device, the Controller driver must clear this bit. In host mode, the PHY can be put into Low Power Suspend Clock Disable when the downstream device has been put into suspend mode or when no downstream device is connected. Low Power Clock Disable is completely under the control of software. 0 = DISABLE 1 = ENABLE
21:20	RW	0x0	LPMX: Auto LPM set. Default = 00b. This bit field is valid during Host Mode Only. For Device Mode, this is reserved. Value Meaning 00b Disables auto LPM. 01b If there is no activity for a certain number of SOFs the controller will send an LPM token, enter in suspend and issue a port change interrupt. 10b Same as above but without issuing an interrupt. 11b Reserved for futures LPM enhancements. The detection of no activity is based on the absence of response from the downstream device. Because of this limitation this feature should not be used if ISO OUT endpoints are being used (as no handshake is expected).
17	RW	0x0	ASUS: Auto Low Power. Default = 0b. This bit field is valid during Device Mode Only. In Host Mode it is part of ELPM field. This bit is used to control the auto low power feature. If set, the auto low power feature will be enabled and every time the port enters the suspend state, it will also enter the low power state, disabling the transceiver clock. The behavior will be the same as if the PHCD bit was enabled (in fact this bit will be set to '1' as soon as low power mode is enabled). 0 = DISABLE 1 = ENABLE
19:16	RW	0x0	EPLPM: Endpoint for LPM token. Default = 0000b. This bit field is valid during Host Mode only. For Device Mode, bits [19:18] are reserved and bit 17 is ASUS, while bit 16 is STL. This sets the endpoint number to which the LPM token will be sent. It will be directly mapped to the ENDP field in the LPM token with the EXT PID.

Bit	R/W	Reset	Description
16	RW	0x0	STL: STALL reply to LPM token. Default = 0b. This bit field is valid during Device Mode only. In Host Mode it is a part of ELPM field. When this bit is set to '1', the Controller will reply always with STALL to all incoming LPM tokens. This bit overrides the LPM NYET bit (NYT). 0 = DISABLE 1 = ENABLE
15:12	RW	0x0	LPMFRM: Auto LPM SOF Threshold. Default = 0000b. This bit field is valid during Host Mode only. For Device Mode, this is reserved. This holds the SOF counter threshold. When the number of SOF with no activity in between reaches this threshold and the auto LPM is enabled, an LPM token will be sent and the port will enter in suspend state. The SOF counter for this threshold is incremented each 125 µs, even if the port is not in HS operation.
11:1	RO	X	BA: BmAttributes . Default = 00000000000b. This holds the bmAttributes field of the LPM sub-token received, after the EXT PID token.
0	RW	0x0	NYT_ASUS: NYET reply to LPM token. Default = 0b. This bit is NYT during Device Mode. Details follow: When this bit is to '1', the device controller will NYET all the LPM tokens. When this bit is set to '0', the Controller will ACK all the LPM tokens if the STALL bit (STL) is also set to '0'. This bit is ASUS in Host Mode. Details follow: This bit is used to control the auto low power feature. If set, the auto low power feature will be enabled and every time the port enters in suspend state it will also enter in low power state, disabling the transceiver clock. The behavior will be the same as if the PHCD bit was enabled (in fact this bit will be set to '1' as soon as low power mode is enabled). 0 = DISABLE 1 = ENABLE

19.13.8.31 USB2_CONTROLLER_2_USB2D_OTGSC_0

USB2D On-The-Go (OTG) Status and Control Register

Offset: 0x1f4 | Read/Write: R/W | Reset: 0bx0000000x0000000xxxxxxxxxx100x00

Bit	R/W	Reset	Description
30	RW	0x0	DPIE: Data Pulse Interrupt Enable. Setting this bit enables the Data pulse interrupt. 0 = DISABLE 1 = ENABLE
29	RW	0x0	ONEMSE: 1 millisecond timer Interrupt enable. Setting this bit enables the 1 millisecond timer interrupt. 0 = DISABLE 1 = ENABLE
28	RW	0x0	BSEIE: B Session End Interrupt Enable. Setting this bit enables the B session end interrupt 0 = DISABLE 1 = ENABLE
27	RW	0x0	BSVIE: B Session Valid Interrupt Enable. Setting this bit enables the B session valid interrupt 0 = DISABLE 1 = ENABLE
26	RW	0x0	ASVIE: A Session Valid Interrupt Enable. Setting this bit enables the A session valid interrupt 0 = DISABLE 1 = ENABLE
25	RW	0x0	AVVIE: A VBus Valid Interrupt Enable. Setting this bit enables the A VBus valid interrupt 0 = DISABLE 1 = ENABLE
24	RW	0x0	IDIE: USB ID Interrupt Enable. Setting this bit enables the USB ID interrupt 0 = DISABLE 1 = ENABLE
22	RW	0x0	DPIS: Data Pulse Interrupt Status. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USBMODE.CM = Host (11) and PORTSC(0).PortPower = Off (0). Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET

Bit	R/W	Reset	Description
21	RW	0x0	ONEMSS: 1 millisecond timer Interrupt Status: This bit is set once every millisecond. Software writes a 1 to clear it. 0 = INT_CLEAR 1 = INT_SET
20	RW	0x0	BSEIS: B Session End Interrupt Status. This bit is set when VBus has fallen below the B session end threshold. Software writes a 1 to clear this bit . 0 = INT_CLEAR 1 = INT_SET
19	RW	0x0	BSVIS: B Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the B session valid threshold (0.8 VDC). Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
18	RW	0x0	ASVIS: A Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the A session valid threshold (0.8 VDC). Software writes a one to clear this bit. 0 = INT_CLEAR 1 = INT_SET
17	RW	0x0	AVVIS: A VBus Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the VBus valid threshold (4.4 VDC) on an A device. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
16	RW	0x0	IDIS: USB ID Interrupt Status. This bit is set when a change on the ID input has been detected. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
14	RO	X	DPS: Data Bus Pulsing Status. A 1 indicates data bus pulsing is being detected on the port. 0 = STS_CLEAR 1 = STS_SET
13	RO	X	ONEMST: 1 millisecond timer toggle. This bit toggles once per millisecond 0 = STS_CLEAR 1 = STS_SET
12	RO	X	BSE: B session End. Indicates VBus is below the B session end threshold 0 = STS_CLEAR 1 = STS_SET
11	RO	X	BSV: B Session Valid. Indicates VBus is above the B session valid threshold 0 = STS_CLEAR 1 = STS_SET
10	RO	X	ASV: A Session Valid. Indicates VBus is above the A session valid threshold 0 = STS_CLEAR 1 = STS_SET
9	RO	X	AVV: A VBus Valid. Indicates VBus is above the A VBus valid threshold 0 = STS_CLEAR 1 = STS_SET
8	RO	X	ID: USB ID: 0 = A-device 1 = B-device 0 = A_DEV 1 = B_DEV
5	RW	0x1	IDPU: USB ID Pullup 0 = CLEAR 1 = SET
4	RW	0x0	DP: Data Pulsing. Setting this bit causes the pull-up on DP to be asserted for data pulsing during SRP. 0 = NO_DATA_PULSE 1 = DATA_PULSE
3	RW	0x0	OT: OTG Termination. This bit must be set when the OTG device is in device mode, this controls the pulldown on DM. 0 = NO_OTG_TERM 1 = OTG_TERM

Bit	R/W	Reset	Description
1	RW	0x0	VC: VBUS Charge. Setting this bit causes the VBus line to be charged. This is used for VBus pulsing during SRP. 0 = NO_VBUS_CHRG 1 = VBUS_CHRG
0	RW	0x0	VD: VBUS_Discharge. Read/write. Setting this bit causes Vbus to discharge through a resistor. 0 = NO_VBUS_DISCHRG 1 = VBUS_DISCHRG

19.13.8.32 USB2_CONTROLLER_2_USB2D_USBMODE_0

USB2D USB Device Mode Register

Offset: 0x1f8 | Read/Write: R/W | Reset: 0b00000000000000000xxxxxxxxx0xxxxx

Bit	R/W	Reset	Description
31:16	RW	0x0	ALPDD: Auto Low Power While Disconnect Delay, Used when the ALPD field of the HOSTPCx register is set. Defines the delay between disconnect detection and entering in low power mode. The delay is <this register value>*64*100/3 in milliseconds. The maximum value is 65535*64*100/3 = 139,808 ms. The minimum value is 0. Only used in host mode.
15	RW	0x0	SRT: Shorten USB Reset Time. Software should never set this bit to 1.
5	RW	0x0	VBPS: VBUS Power Select This can be used by logic that selects between an on-chip Vbus power source (charge pump) and an off-chip source in systems when both are available. Only to be used in Host Mode. No functionality is implemented for this, so software should not use this.
4	RO	X	SDIS: Stream disable: 1 Streaming is disabled - helpful to avoid overruns/underruns when system load is too high. 0 = STREAM_ENABLE 1 = STREAM_DISABLE
3	RO	X	SLOM: Setup Lockout Mode: In device mode, this bit controls the behavior of the setup lockout mechanism. 0 - Setup lockout is ON (default) 1 - Setup lockout is OFF. Firmware requires the use of setup tripwire semaphore in USB2D_USBCMD register. 0 = LOCKOUT_ON 1 = LOCKOUT_OFF
2	RO	X	ES: Endian Select: Note: For this implementation, this should be always set to 0 (little endian). 0 = LITTLE_ENDIAN 1 = RESERVED
1:0	RO	X	CM: Controller Mode: The controller mode will default to an idle state and will need to be initialized to the desired operating mode after reset. This register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the RESET bit in the USBCMD register before reprogramming this register. 00 = Idle [Default] 01 = Reserved 10 = Device Controller 11 = Host Controller. 0 = IDLE 1 = RESERVED 2 = DEVICE_MODE 3 = HOST_MODE

19.13.8.33 USB2_CONTROLLER_2_USB2D_ENDPTNAK_0

USB2D Endpoint NAK Register

Offset: 0x200 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	EPTN: TX Endpoint NAK R/WC. Each TX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received IN token for the corresponding endpoint. 0 = CLEAR 1 = SET
15:0	0x0	EPRN: RX Endpoint NAK R/WC. Each RX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received OUT or PING token for the corresponding endpoint. 0 = CLEAR 1 = SET

19.13.8.34 USB2_CONTROLLER_2_USB2D_ENDPTNAK_ENABLE_0

USB2D Endpoint NAK Enable Register

Offset: 0x204 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	EPTNE: TX Endpoint NAK Enable R/W. Each bit is an enable bit for the corresponding TX Endpoint NAK bit. If this bit is set and the corresponding TX Endpoint NAK bit is set, the NAK Interrupt bit is set. 0 = DISABLE 1 = ENABLE
15:0	0x0	EPRNE: RX Endpoint NAK Enable R/W. Each bit is an enable bit for the corresponding RX Endpoint NAK bit. If this bit is set and the corresponding RX Endpoint NAK bit is set, the NAK Interrupt bit is set. 0 = DISABLE 1 = ENABLE

19.13.8.35 USB2_CONTROLLER_2_USB2D_ENDPTSETUPSTAT_0

USB2D Endpoint Setup Status Register

Offset: 0x208 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
15	0x0	ENDPTSETUPSTAT15: Endpoint 15 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
14	0x0	ENDPTSETUPSTAT14: Endpoint 14 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
13	0x0	ENDPTSETUPSTAT13: Endpoint 13 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD

Bit	Reset	Description
12	0x0	ENDPTSETUPSTAT12: Endpoint 12 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
11	0x0	ENDPTSETUPSTAT11: Endpoint 11 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
10	0x0	ENDPTSETUPSTAT10: Endpoint 10 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
9	0x0	ENDPTSETUPSTAT9: Endpoint 9 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
8	0x0	ENDPTSETUPSTAT8: Endpoint 8 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
7	0x0	ENDPTSETUPSTAT7: Endpoint 7 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
6	0x0	ENDPTSETUPSTAT6: Endpoint 6 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
5	0x0	ENDPTSETUPSTAT5: Endpoint 5 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD

Bit	Reset	Description
4	0x0	ENDPTSETUPSTAT4: Endpoint 4 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
3	0x0	ENDPTSETUPSTAT3: Endpoint 3 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
2	0x0	ENDPTSETUPSTAT2: Endpoint 2 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
1	0x0	ENDPTSETUPSTAT1: Endpoint 1 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
0	0x0	ENDPTSETUPSTAT0: Endpoint 0 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD

19.13.8.36 USB2_CONTROLLER_2_USB2D_ENDPTPRIME_0

USB2D Endpoint Initialization Register

Offset: 0x20c | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	PETB15: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
30	0x0	PETB14: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
29	0x0	PETB13: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
28	0x0	PETB12: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
27	0x0	PETB11: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
26	0x0	PETB10: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
25	0x0	PETB9: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
24	0x0	PETB8: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
23	0x0	PETB7: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
22	0x0	PETB6: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
21	0x0	PETB5: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
20	0x0	PETB4: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
19	0x0	PETB3: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
18	0x0	PETB2: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
17	0x0	PETB1: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
16	0x0	PETB0: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
15	0x0	PERB15: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
14	0x0	PERB14: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
13	0x0	PERB13: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
12	0x0	PERB12: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
11	0x0	PERB11: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
10	0x0	PERB10: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
9	0x0	PERB9: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
8	0x0	PERB8: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
7	0x0	PERB7: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
6	0x0	PERB6: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
5	0x0	PERB5: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
4	0x0	PERB4: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
3	0x0	PERB3: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
2	0x0	PERB2: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
1	0x0	PERB1: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
0	0x0	PERB0: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

19.13.8.37 USB2_CONTROLLER_2_USB2D_ENDPTFLUSH_0

USB2D Endpoint De-Initialization Register

Offset: 0x210 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	FETB15: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
30	0x0	FETB14: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
29	0x0	FETB13: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
28	0x0	FETB12: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
27	0x0	FETB11: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
26	0x0	FETB10: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
25	0x0	FETB9: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
24	0x0	FETB8: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
23	0x0	FETB7: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
22	0x0	FETB6: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
21	0x0	FETB5: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
20	0x0	FETB4: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
19	0x0	FETB3: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
18	0x0	FETB2: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
17	0x0	FETB1: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
16	0x0	FETB0: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
15	0x0	FERB15: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
14	0x0	FERB14: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
13	0x0	FERB13: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
12	0x0	FERB12: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
11	0x0	FERB11: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
10	0x0	FERB10: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
9	0x0	FERB9: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
8	0x0	FERB8: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
7	0x0	FERB7: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
6	0x0	FERB6: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
5	0x0	FERB5: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
4	0x0	FERB4: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
3	0x0	FERB3: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
2	0x0	FERB2: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
1	0x0	FERB1: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
0	0x0	FERB0: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

19.13.8.38 USB2_CONTROLLER_2_USB2D_ENDPTSTATUS_0

USB2D Endpoint Status Register

Offset: 0x214 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	ETBR15: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
30	X	ETBR14: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
29	X	ETBR13: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
28	X	ETBR12: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
27	X	ETBR11: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
26	X	ETBR10: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
25	X	ETBR9: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
24	X	ETBR8: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
23	X	ETBR7: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
22	X	ETBR6: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
21	X	ETBR5: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
20	X	ETBR4: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
19	X	ETBR3: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
18	X	ETBR2: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
17	X	ETBR1: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
16	X	ETBR0: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
15	X	ERBR15: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
14	X	ERBR14: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
13	X	ERBR13: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
12	X	ERBR12: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
11	X	ERBR11: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
10	X	ERBR10: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
9	X	ERBR9: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
8	X	ERBR8: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
7	X	ERBR7: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
6	X	ERBR6: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
5	X	ERBR5: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
4	X	ERBR4: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
3	X	ERBR3: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
2	X	ERBR2: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
1	X	ERBR1: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
0	X	ERBR0: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

19.13.8.39 USB2_CONTROLLER_2_USB2D_ENDPTCOMPLETE_0

USB2D Endpoint Complete Register

Offset: 0x218 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ETCE15: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
30	0x0	ETCE14: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
29	0x0	ETCE13: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
28	0x0	ETCE12: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
27	0x0	ETCE11: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
26	0x0	ETCE10: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
25	0x0	ETCE9: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
24	0x0	ETCE8: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
23	0x0	ETCE7: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
22	0x0	ETCE6: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
21	0x0	ETCE5: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
20	0x0	ETCE4: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
19	0x0	ETCE3: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
18	0x0	ETCE2: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
17	0x0	ETCE1: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
16	0x0	ETCE0: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
15	0x0	ERCE15: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
14	0x0	ERCE14: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
13	0x0	ERCE13: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
12	0x0	ERCE12: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
11	0x0	ERCE11: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
10	0x0	ERCE10: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
9	0x0	ERCE9: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
8	0x0	ERCE8: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
7	0x0	ERCE7: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
6	0x0	ERCE6: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
5	0x0	ERCE5: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
4	0x0	ERCE4: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
3	0x0	ERCE3: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
2	0x0	ERCE2: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
1	0x0	ERCE1: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
0	0x0	ERCE0: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

19.13.8.40 USB2_CONTROLLER_2_USB2D_ENDPTCTRL0_0

USB2D Endpoint Control 0 Register

Offset: 0x21c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23	X	TXE: TX Endpoint Enable. Endpoint 0 is always enabled. 0 = DISABLE 1 = ENABLE
19:18	X	TXT: TX Endpoint Type. Endpoint0 is fixed as a Control Endpoint. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
16	X	TXS: TX Endpoint Stall: Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0 = EP_OK 1 = EP_STALL
7	X	RXE: RX Endpoint Enable. Endpoint 0 is always enabled. 0 = DISABLE 1 = ENABLE
3:2	X	RXT: RX Endpoint Type. Endpoint 0 is fixed as a Control Endpoint. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
0	X	RXS: RX Endpoint Stall: Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0 = EP_OK 1 = EP_STALL

19.13.8.41 USB2_CONTROLLER_2_USB2D_ENDPTCTRLn_0

USB2D Endpoint Control n Register

USB2D Endpoint Control 1 through 15 registers have the same field definitions and reset value. In the register offset, the value n is the register number (1 through 15).

Offset: 0x220 + (n – 1) * 0x04 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL

Bit	R/W	Reset	Description
			1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above, 0 = EP_OK 1 = EP_STALL

19.13.9 USB3 Controller Interface Registers

These are used to generate the actual RTL registers for the USB3 controller interface.

19.13.9.1 USB3_IF_USB_SUSP_CTRL_0

USB Suspend Control Register

This register controls the suspend and resume behavior of the USB controller/PHY.

Offset: 0x400 | Read/Write: R/W | Reset: 0bxxxxx00xxxxx000001001000xx000000

Bit	R/W	Reset	Description
26	RW	0x0	FAST_WAKEUP_RESP: Enable Fast Response from UTMIP PHY for a Remote Wakeup request from device. This is used only for cases where wakeup response needs to be within 1 ms of spec. Used for Host mode ONLY. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
25	RW	0x0	UTMIP_SUSPL1_SET: Enable SuspendL1 for UTMIP PHY. Enabling this will only cut off clocks to the UTMIP logic. The USB PLLs, PllU and UTMIP PLL will still be running. 0 = DISABLE 1 = ENABLE
19	RW	0x0	UHSIC_PHY_ENB: Set this to 0. 0 = DISABLE 1 = ENABLE
18:16	RW	0x0	USB_WAKEUP_DEBOUNCE_COUNT: USB PHY wakeup debounce counter. The USB will debounce any wakeup event by the number of clocks programmed in this counter. A value of 0 results in no debounce.
15	RW	0x0	ICUSB_MOD_CLK_ENB: ICUSB module clock enable. Enables transceiver clock to USB controller when in ICUSB mode. After setting ICUSB_PHY_ENB, software needs to wait until PLLU output is stable before setting ICUSB_MOD_CLK_ENB to ENABLE. This will be reset to DISABLE whenever ICUSB is in suspend. Software needs to enable it after ICUSB comes out of suspend after waiting for PLLU output to be stable again. 0 = DISABLE 1 = ENABLE
14	RW	0x1	UHSIC_RESET: Reset going to UHSIC PHY (active high). This should be set to 1 whenever programming the UHSIC config registers. It should be cleared to 0 after the programming of UHSIC config registers is done. UHSIC config registers should be programmed only once before doing any transactions on UHSIC. The UHSIC PHY registers should be programmed while the UHSIC is in reset. 0 = DISABLE 1 = ENABLE
13	RW	0x0	ICUSB_PHY_ENB: Enable ICUSB PHY mode. Setting this will enable the PLLU output clock when ICUSB is not in suspend mode. 0 = DISABLE 1 = ENABLE
12	RW	0x0	UTMIP_PHY_ENB: Enable UTMIP PHY mode. Set this to 1 if using UTMIP PHY. Otherwise set this to 0. 0 = DISABLE 1 = ENABLE
11	RW	0x1	UTMIP_RESET: Reset going to UTMIP PHY (active high). This should be set to 1 whenever programming the UTMIP config registers. It should be cleared to 0 after the programming of UTMIP config registers is done. UTMIP config registers should be programmed only once before doing any transactions on USB. The UTMIP PHY registers should be programmed while UTMIP is in reset. 0 = DISABLE 1 = ENABLE
10	RW	0x0	USB_SUSP_POL: Polarity of the suspend signal going to the USB PHY. 0 = Active low (default), 1 = Active high. This should not be changed by software. 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
9	RW	0x0	USB_PHY_CLK_VALID_INT_ENB: USB PHY clock valid interrupt enable. If this bit is enabled, interrupt is generated whenever USB clocks are resumed from a suspend. 0 = DISABLE 1 = ENABLE
8	RO	0x0	USB_PHY_CLK_VALID_INT_STS: USB PHY clock valid interrupt status. This bit is set whenever the USB PHY clock is woken up from suspend. Software must write a 1 to clear this bit. 0 = UNSET 1 = SET
7	RO	X	USB_PHY_CLK_VALID: USB PHY clock valid status. This bit indicates whether the USB PHY is generating a valid clock to the USB controller. If USB PHY clock is running, this bit is set to 1, else it is set to 0. 0 = UNSET 1 = SET
6	RO	X	USB_CLKEN: USB AHB clock enable status. Indicates whether the AHB clock to the USB controller is enabled or not. If AHB clock to USB controller is enabled, this bit is set to 1, else it is

Bit	R/W	Reset	Description
			set to 0. NOTE: Even when this is set to 0, all essential blocks that are required to resume USB clocks from suspend will be active and their AHB clock will not be suspended. 0 = UNSET 1 = SET
5	RW	0x0	USB_SUSP_CLR: Suspend Clear. Software must write a 1 to this bit to bring the PHY out of suspend mode. This is used when the software stops the PHY clock during suspend and then wants to initiate a resume. Software should also write 0 to clear it. NOTE: It is required that software generate a positive pulse on this bit to guarantee proper operation. 0 = UNSET 1 = SET
4	RW	0x0	USB_WAKE_ON_DISCON_EN_DEV: Wake on Disconnect Enable (device mode). When enabled (1), USB device will wake up from suspend on a disconnect event. This is only valid when USB controller is in device mode, it is not applicable when USB controller is in host mode. 0 = DISABLE 1 = ENABLE
3	RW	0x0	USB_WAKE_ON_CNNT_EN_DEV: Wake on Connect Enable (device mode). When enabled (1), USB device will wake up from suspend on a connect event. This is only valid when USB controller is in device mode, it is not applicable when USB controller is in host mode. 0 = DISABLE 1 = ENABLE
2	RW	0x0	USB_WAKE_ON_RESUME_EN: Wake on resume enable. If this bit is enabled, USB will wake up from suspend whenever a resume event is detected on USB. This is valid for both USB device and USB host modes. 0 = DISABLE 1 = ENABLE
1	RW	0x0	USB_WAKEUP_INT_ENB: USB wakeup interrupt enable. If this bit is enabled, interrupt is generated whenever USB wakeup event is generated. 0 = DISABLE 1 = ENABLE
0	RO	0x0	USB_WAKEUP_INT_STS: USB wakeup interrupt status. This bit is set whenever USB wakes up from suspend (a wakeup event is generated). Software must write a 1 to clear this bit. Note that during the wakeup sequence, PHY clocks will be resumed from suspend. Software can check when the PHY clocks are resumed by reading the bit USB_PHY_CLK_VALID. There is also a separate interrupt generated when PHY clock is resumed if USB_PHY_CLK_VALID_INT_EN is set. During the wakeup sequence, first USB_WAKEUP_INT_STS will be set, and it will take some time for the PHY clock to resume, which can be detected by checking USB_PHY_CLK_VALID. 0 = UNSET 1 = SET

19.13.9.2 USB3_IF_USB_PHY_VBUS_SENSORS_0

USB PHY VBUS SENSORS Control Register

This register controls the OTG VBUS sensors in the USB PHY. There are 4 VBUS sensors:

- A_VBUS_VLD
- A_SESS_VLD
- B_SESS_VLD
- B_SESS_END

The debounced status of each sensor can be read from the corresponding _STS bit field of the sensor in this register. The _CHG_DET field is set to 1 whenever a change is detected in the value of the _STS bit field of the corresponding sensor. If _INT_EN is set, then an interrupt is generated to the processor. This interrupt can be routed to CPU/AVP by appropriately writing the USBD bits in the interrupt controller registers.

In case software wants to override the value for a sensor, it can set the corresponding _SW_EN to 1, and set the corresponding sensor _SW_VALUE to 1 or 0 as per the requirement.

There are two debouncers for each sensor - DEBOUNCE_A and DEBOUNCE_B. The debounce values for them are controlled by the register UTMIP_DEBOUNCE_CFG0, fields UTMIP_BIAS_DEBOUNCE_A and UTMIP_BIAS_DEBOUNCE_B. For each sensor, we can select whether to use DEBOUNCE_A or DEBOUNCE_B by setting the field _DEB_SEL_B to the appropriate value (SEL_A or SEL_B).

Note: Do not set either UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.

Offset: 0x404 | Read/Write: R/W | Reset: 0bx0000x00x0000x00x0000x00x0000x00

Bit	R/W	Reset	Description
30	RW	0x0	A_VBUS_VLD_WAKEUP_EN: A_VBUS_VLD wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on A_VBUS_VLD. 0 = DISABLE 1 = ENABLE
29	RW	0x0	A_VBUS_VLD_DEB_SEL_B: A_VBUS_VLD debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
28	RW	0x0	A_VBUS_VLD_SW_VALUE: A_VBUS_VLD software value. Software should write the appropriate value (1/0) to set/unset the A_VBUS_VLD status. This is only valid when A_VBUS_VLD_SW_EN is set. 0 = UNSET 1 = SET
27	RW	0x0	A_VBUS_VLD_SW_EN: A_VBUS_VLD software enable. Enable Software Controlled A_VBUS_VLD. Software sets this bit to drive the value in A_VBUS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
26	RO	X	A_VBUS_VLD_STS: A_VBUS_VLD status. This is set to 1 whenever A_VBUS_VLD sensor output is 1. 0 = UNSET 1 = SET
25	RO	0x0	A_VBUS_VLD_CHG_DET: A_VBUS_VLD change detect. This field is set by hardware whenever a change is detected in the value of A_VBUS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET
24	RW	0x0	A_VBUS_VLD_INT_EN: A_VBUS_VLD interrupt enable. If this field is set to 1, an interrupt is generated whenever A_VBUS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
22	RW	0x0	A_SESS_VLD_WAKEUP_EN: A_SESS_VLD wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on A_SESS_VLD. 0 = DISABLE 1 = ENABLE
21	RW	0x0	A_SESS_VLD_DEB_SEL_B: A_SESS_VLD debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
20	RW	0x0	A_SESS_VLD_SW_VALUE: A_SESS_VLD software value. Software should write the appropriate value (1/0) to set/unset the A_SESS_VLD status. This is only valid when A_SESS_VLD_SW_EN is set. 0 = UNSET 1 = SET
19	RW	0x0	A_SESS_VLD_SW_EN: A_SESS_VLD software enable. Enable Software Controlled A_SESS_VLD. Software sets this bit to drive the value in A_SESS_VLD_SW_VALUE to the USB controller. 0 = DISABLE

Bit	R/W	Reset	Description
			1 = ENABLE
18	RO	X	A_SESS_VLD_STS: A_SESS_VLD status. This is set to 1 whenever A_SESS_VLD sensor output is 1. 0 = UNSET 1 = SET
17	RO	0x0	A_SESS_VLD_CHG_DET: A_SESS_VLD change detect. This field is set by hardware whenever a change is detected in the value of A_SESS_VLD. Software writes a 1 to clear it. 0 = UNSET 1 = SET
16	RW	0x0	A_SESS_VLD_INT_EN: A_SESS_VLD interrupt enable. If this field is set to 1, an interrupt is generated whenever A_SESS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
14	RW	0x0	B_SESS_VLD_WAKEUP_EN: B_SESS_VLD wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on B_SESS_VLD. 0 = DISABLE 1 = ENABLE
13	RW	0x0	B_SESS_VLD_DEB_SEL_B: B_SESS_VLD debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
12	RW	0x0	B_SESS_VLD_SW_VALUE: B_SESS_VLD software value. Software should write the appropriate value (1/0) to set/unset the B_SESS_VLD status. This is only valid when B_SESS_VLD_SW_EN is set. 0 = UNSET 1 = SET
11	RW	0x0	B_SESS_VLD_SW_EN: B_SESS_VLD software enable. Enable Software Controlled B_SESS_VLD. Software sets this bit to drive the value in B_SESS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
10	RO	X	B_SESS_VLD_STS: B_SESS_VLD status. This is set to 1 whenever B_SESS_VLD sensor output is 1. 0 = UNSET 1 = SET
9	RO	0x0	B_SESS_VLD_CHG_DET: B_SESS_VLD change detect. This field is set by hardware whenever a change is detected in the value of B_SESS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET
8	RW	0x0	B_SESS_VLD_INT_EN: B_SESS_VLD interrupt enable. If this field is set to 1, an interrupt is generated whenever B_SESS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
6	RW	0x0	B_SESS_END_WAKEUP_EN: B_SESS_END wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on B_SESS_END. 0 = DISABLE 1 = ENABLE
5	RW	0x0	B_SESS_END_DEB_SEL_B: B_SESS_END debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
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

Bit	R/W	Reset	Description
3	RW	0x0	B_SESS_END_SW_EN: Enable Software Controlled B_SESS_END. Software sets this bit to drive the value in B_SESS_END_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
2	RO	X	B_SESS_END_STS: B_SESS_END status. This is set to 1 whenever the B_SESS_END sensor output is 1. 0 = UNSET 1 = SET
1	RO	0x0	B_SESS_END_CHG_DET: B_SESS_END change detect. This field is set by hardware whenever a change is detected in the value of B_SESS_END. software writes a 1 to clear it 0 = UNSET 1 = SET
0	RW	0x0	B_SESS_END_INT_EN: B_SESS_END interrupt enable. If this field is set to 1, an interrupt is generated whenever B_SESS_END_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE

19.13.9.3 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. The following sensors are in this register:

- VBUS_WAKEUP
- ID
- VDAT_DET
- VDCD_DET

The debounced status of each sensor can be read from the corresponding _STS bit field of the sensor in this register. The _CHG_DET field is set to 1 whenever a change is detected in the value of the _STS bit field of the corresponding sensor. If _INT_EN is set, then an interrupt is generated to the processor. This interrupt can be routed to CPU/AVP by appropriately writing the USBD bits in the interrupt controller registers.

In case software wants to override the value for a sensor, it can set the corresponding _SW_EN to 1, and set the corresponding sensor _SW_VALUE to 1 or 0 as per the requirement.

There are two debouncers for each sensor - DEBOUNCE_A and DEBOUNCE_B. The debounce values for them are controlled by the register UTMIP_DEBOUNCE_CFG0, fields UTMIP_BIAS_DEBOUNCE_A and UTMIP_BIAS_DEBOUNCE_B. For each sensor, we can select whether to use DEBOUNCE_A or DEBOUNCE_B by setting the field _DEB_SEL_B to the appropriate value (SEL_A or SEL_B).

Note: Do not set either UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.

There are debouncers for VDAT_DET and VDCD_DET. These use separate debouncers - CHRG_DEBOUNCE_PERIOD_A and CHRG_DEBOUNCE_PERIOD_B. The debounce values for them are controlled by the register UTMIP_CHRG_DEB_CFG0, fields UTMIP_CHRG_DEBOUNCE_PERIOD_A and UTMIP_CHRG_DEBOUNCE_PERIOD_B. For each sensor, we can select whether to use HRG_DEBOUNCE_PERIOD_A or CHRG_DEBOUNCE_PERIOD_B by setting the field _DEB_SEL_B to the appropriate value (SEL_A or SEL_B).

Note: Do not set either UTMIP_CHRG_DEBOUNCE_PERIOD_A or UTMIP_CHRG_DEBOUNCE_PERIOD_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.

Offset: 0x408 | Read/Write: R/W | Reset: 0b00000x00xx000x00xx000x00x1000x00

Bit	R/W	Reset	Description
31	RW	0x0	DIV_DET_EN: Battery charger divider detection enable. This goes to the USB2OTG pad. 0 = DISABLE 1 = ENABLE
30	RW	0x0	VBUS_WAKEUP_WAKEUP_EN: VBUS_WAKEUP wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on VBUS_WAKEUP. 0 = DISABLE 1 = ENABLE
29	RW	0x0	VDCD_DET_DEB_SEL_B: VCDT_DET debounce A/B select. Selects the debounce value from UTMIP_CHRG_DEBOUNCE_PERIOD_A or UTMIP_CHRG_DEBOUNCE_PERIOD_B from the register UTMIP_CHRG_DEB_CFG0. 0 = SEL_A 1 = SEL_B
28	RW	0x0	VDCD_DET_SW_VALUE: VDCD_DET software value. Software should write the appropriate value (1/0) to set/unset the VDCD_DET status. This is only valid when VDCD_DET_SW_EN is set. 0 = UNSET 1 = SET
27	RW	0x0	VDCD_DET_SW_EN: VDCD_DET software enable. Enable Software Controlled VDCD_DET. Software sets this bit to drive the value in VDCD_DET_SW_VALUE to the USB controller 0 = DISABLE 1 = ENABLE
26	RO	X	VDCD_DET_STS: VDCD_DET status. This is set to 1 whenever VDCD_DET sensor output is 1. 0 = UNSET 1 = SET
25	RO	0x0	VDCD_DET_CHG_DET: VDCD_DET change detect. This field is set by hardware whenever a change is detected in the value of VDCD_DET. Software writes a 1 to clear it. 0 = UNSET 1 = SET
24	RW	0x0	VDCD_DET_INT_EN: VDCD_DET interrupt enable. If this field is set to 1, an interrupt is generated whenever VDCD_DET_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
23	RO	X	VOP_DIV2P7_DET:This read-only status bit is from the battery charging divider circuit of USB2OTG pad 0 = UNSET 1 = SET
22	RO	X	VOP_DIV2P0_DET:This read-only status bit is from the battery charging divider circuit of USB2OTG pad 0 = UNSET 1 = SET
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: 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

Bit	R/W	Reset	Description
			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
15	RO	X	VON_DIV2P7_DET: This read-only status bit is from the battery charging divider circuit of USB2OTG pad 0 = UNSET 1 = SET
14	RO	X	VON_DIV2P0_DET: This read-only status bit is from the battery charging divider circuit of USB2OTG pad generated whenever VDAT_DET_CHG_DET is set to 1. 0 = UNSET 1 = SET
13	RW	0x0	VBUS_WAKEUP_DEB_SEL_B: VBUS_WAKEUP debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
12	RW	0x0	VBUS_WAKEUP_SW_VALUE: VBUS wakeup software value. Software should write the appropriate value (1/0) to set/unset the VBUS_WAKEUP status. This is only valid when VBUS_WAKEUP_SW_EN is set. 0 = UNSET 1 = SET
11	RW	0x0	VBUS_WAKEUP_SW_EN: Enable Software Controlled VBUS_WAKEUP. Software sets this bit to drive the value in VBUS_WAKEUP_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
10	RO	X	VBUS_WAKEUP_STS: VBUS wakeup status. This is set to 1 whenever VBUS_WAKEUP sensor output is 1. 0 = UNSET 1 = SET
9	RO	0x0	VBUS_WAKEUP_CHG_DET: VBUS wakeup change detect. This field is set by hardware whenever a change is detected in the value of VBUS_WAKEUP. software writes a 1 to clear it. 0 = UNSET 1 = SET
8	RW	0x0	VBUS_WAKEUP_INT_EN: VBUS wakeup interrupt enable. If this field is set to 1, an interrupt is generated whenever VBUS_WAKEUP_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
7	RO	X	STATIC_GPI: Static GPI status 0 = UNSET 1 = SET
6	RW	0x1	ID_PU: ID pullup enable. Set to 1. 0 = DISABLE 1 = ENABLE
5	RW	0x0	ID_DEB_SEL_B: ID debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
4	RW	0x0	ID_SW_VALUE: ID software value. Software should write the appropriate value (1/0) to set/unset the ID status. This is only valid when ID_SW_EN is set. 0 = UNSET 1 = SET
3	RW	0x0	ID_SW_EN: Enable Software Controlled ID. Software sets this bit to drive the value in ID_SW_VALUE to the USB controller 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
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

19.13.9.4 USB3_IF_USB_PHY_ALT_VBUS_STS_0

USB PHY Alternate VBUS Status Register

Offset: 0x40c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
8	X	VDCD_DET_ALT: VDCD_DET alternate status 0 = UNSET 1 = SET
7	X	VDAT_DET_ALT: VDAT_DET alternate status 0 = UNSET 1 = SET
6	X	A_SESS_VLD_ALT: A_SESS_VLD alternate status 0 = UNSET 1 = SET
5	X	B_SESS_VLD_ALT: B_SESS_VLD alternate status 0 = UNSET 1 = SET
4	X	ID_DIG_ALT: ID alternate status 0 = UNSET 1 = SET
3	X	B_SESS_END_ALT: B_SESS_END alternate status 0 = UNSET 1 = SET
2	X	STATIC_GPI_ALT: Static GPI alternate status 0 = UNSET 1 = SET
1	X	A_VBUS_VLD_ALT: A_VBUS_VLD alternate status 0 = UNSET 1 = SET
0	X	VBUS_WAKEUP_ALT: Vbus wakeup alternate status 0 = UNSET 1 = SET

19.13.9.5 USB3_IF_ICUSB_XCVR_CFG_0

ICUSB Transceiver Configuration Register

Offset: 0x414 | Read/Write: R/W | Reset: 0bxxxxxxxxxxx10000xx0000000xxx1111

Bit	R/W	Reset	Description
31:24	RO	X	ICUSB_CALOUT: ICUSB PHY calibration code

Bit	R/W	Reset	Description
20	RW	0x1	ICUSB_CALOUT_EN: ICUSB PHY auto-calibration enable 0 = DISABLE 1 = ENABLE
19:16	RW	0x0	ICUSB_DRV: ICUSB PHY Drive strength offset
13:8	RW	0x0	ICUSB_SLEW: ICUSB PHY FS/LS slew rate control
7	RW	0x0	ICUSB_SEL_DIFF_RCVR: ICUSB differential receiver select 0 = SINGLE 1 = DIFF
3	RW	0x1	ICUSB_IDDQ: ICUSB PHY IDDQ shutdown mode 0 = NORMAL 1 = OFF
2	RW	0x1	ICUSB_PD_ZI: ICUSB PHY Single-ended receiver power down 0 = NORMAL 1 = OFF
1	RW	0x1	ICUSB_PD_DR: ICUSB PHY Differential receiver power down 0 = NORMAL 1 = OFF
0	RW	0x1	ICUSB_PD_TX: ICUSB PHY Low/full-speed driver power down 0 = NORMAL 1 = OFF

19.13.9.6 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 because device never transmits two packets in a row.

Offset: 0x420 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0010010

Bit	Reset	Description
6:0	0x12	IP_DELAY_TX2TX_HS: HS Tx to Tx inter-packet delay. This is valid only for the UTMIP PHY. Software should not change this.

19.13.9.7 USB3_IF_USB_RSM_DLY_0

USB Controller Resume Signalling Delay

Offset: 0x490 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0110100101111000

Bit	Reset	Description
15:0	0x6978	TIME_TO_RESUME: Send the resume back in number of 60 MHz cycles. Default gives 900 µs delay. Only applicable in host mode.

19.13.9.8 USB3_IF_ICUSB_PADCTLS_0

ICUSB Pad Controls Config

Offset: 0x494 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx000000000

Bit	Reset	Description
8	0x0	ICUSB_RPU2_EN: Config RPU2_EN state. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	0x0	ICUSB_DISABLE_PULLUP_DP: Force DP pull-up inactive. (Overrides FORCE_PULLUP_DP.)
6	0x0	ICUSB_DISABLE_PULLUP_DM: Force DM pull-up inactive. (Overrides FORCE_PULLUP_DM.)
5	0x0	ICUSB_DISABLE_PULLDN_DP: Force DP pull-down inactive. (Overrides FORCE_PULLDN_DP.)
4	0x0	ICUSB_DISABLE_PULLDN_DM: Force DM pull-down inactive. (Overrides FORCE_PULLDN_DM.)
3	0x0	ICUSB_FORCE_PULLUP_DP: Force DP pull-up active.
2	0x0	ICUSB_FORCE_PULLUP_DM: Force DM pull-up active.
1	0x0	ICUSB_FORCE_PULLDN_DP: Force DP pull-down active.
0	0x0	ICUSB_FORCE_PULLDN_DM: Force DM pull-down active.

19.13.9.9 USB3_IF_SPARE_0

Spare Register

Offset: 0x498 | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:16	0xffff	SPARE_HI: Spare register bits, defaulted to 1. SPARE_HI[0] - usb_port_reset_fix_en. Enable port-reset fix SPARE_HI[1] - hsic_conn_det_feature_enable. enable HSIC connect detection
15:0	0x0	SPARE_LO: Spare register bits, defaulted to 0. SPARE_LO[4] - usb_hs_rsm_eop_fix_en. Enable HS_RESUME EOP fix SPARE_LO[5] - usb_port_suspend_fix_en

19.13.9.10 USB3_IF_USB1_NEW_CONTROL_0

USB Coherency and Memory Alignment Controls

Offset: 0x4c0 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxx00001111xxxxxx00

Bit	Reset	Description
15:8	0xf	REQUEST_EXPIRY_COUNTER: Time to wait for coalescing the request.
1	0x0	MEM_ALIGNMENT_MUX_EN: Mux to select between Tegra 3 style (0) and Tegra K1 style (1) DMA request generation mechanism 0 = DISABLE 1 = ENABLE
0	0x0	COHERENCY_EN: Enable fence mechanism. 0 = DISABLE 1 = ENABLE

19.13.10 XUSB Host Registers

19.13.10.1 XUSB_HOST_AXI_BAR0_SZ_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxxxxxxxx00000000000000001000)

Bit	Reset	Description
19:0	0x8	AXI_BAR0_SIZE: The size of the address range associated with BARi is in 4K increments. Value of 0 signifies BARi is not used.

19.13.10.2 XUSB_HOST_AXI_BAR1_SZ_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	AXI_BAR1_SIZE: The size of the address range associated with BARi is in 4K increments. Value of 0 signifies BARi is not used.

19.13.10.3 XUSB_HOST_AXI_BAR2_SZ_0

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	AXI_BAR2_SIZE: The size of the address range associated with BARi is in 4K increments. Value of 0 signifies BARi is not used.

19.13.10.4 XUSB_HOST_AXI_BAR3_SZ_0

Offset: 0xc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	AXI_BAR3_SIZE: The size of the address range associated with BARi is in 4K increments. Value of 0 signifies BARi is not used.

19.13.10.5 XUSB_HOST_AXI_BAR0_START_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x70090000 (0b01110000000010010000xxxxxxxxxxxx)

Bit	Reset	Description
31:12	0x70090	AXI_BAR0_START: The start of the AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

19.13.10.6 XUSB_HOST_AXI_BAR1_START_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxxx)

Bit	Reset	Description
31:12	0x0	AXI_BAR1_START: The start of AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

19.13.10.7 XUSB_HOST_AXI_BAR2_START_0

Offset: 0x48 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxxx)

Bit	Reset	Description
31:12	0x0	AXI_BAR2_START: The start of the AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

19.13.10.8 XUSB_HOST_AXI_BAR3_START_0

Offset: 0x4c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxxx)

Bit	Reset	Description
31:12	0x0	AXI_BAR3_START: The start of the AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

19.13.10.9 XUSB_HOST_FPCI_BAR0_0

Offset: 0x80 | Read/Write: R/W | Reset: 0x00700901 (0b00000000011100000000010010000xxx1)

Bit	Reset	Description
31:4	0x70090	FPCI_BAR0_START: The start of the FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a 12-bit left shift of the value of this register.
0	0x1	FPCI_BAR0_ACCESS_TYPE: Indicates if the address region is memory mapped versus configuration or I/O space. 0 = memory-mapped access (PW only) 1 = I/O or config access (NPW only)

19.13.10.10 XUSB_HOST_FPCI_BAR1_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000000000000000xxx1)

Bit	Reset	Description
31:4	0x0	FPCI_BAR1_START: The start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a 12-bit left shift of the value of this register.
0	0x1	FPCI_BAR1_ACCESS_TYPE: Indicates if the address region is memory mapped versus configuration or I/O space. 0 = Memory-mapped access (PW only) 1 = I/O or config access (NPW only)

19.13.10.11 XUSB_HOST_FPCI_BAR2_0

Offset: 0x88 | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000000000000000xxx1)

Bit	Reset	Description
31:4	0x0	FPCI_BAR2_START: The start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a 12-bit left shift of the value of this register.
0	0x1	FPCI_BAR2_ACCESS_TYPE: Indicates if the address region is memory mapped versus configuration or I/O space. 0 = Memory-mapped access (PW only) 1 = I/O or config access (NPW only)

19.13.10.12 XUSB_HOST_FPCI_BAR3_0

Offset: 0x8c | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000000000000000xxx1)

Bit	Reset	Description
31:4	0x0	FPCI_BAR3_START: The start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a 12-bit left shift of the value of this register.
0	0x1	FPCI_BAR3_ACCESS_TYPE: Indicates if the address region is memory mapped versus configuration or I/O space. 0 = Memory-mapped access (PW only) 1 = I/O or config access (NPW only)

19.13.10.13 XUSB_HOST_MSI_BAR_SZ_0

MSI BAR SIZE

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	MSI_BAR_SIZE: The size of the address range associated with MSI BAR is in 4K increments. Value of 0 signifies BAR is not used.

19.13.10.14 XUSB_HOST_MSI_AXI_BAR_ST_0

MSI AXI BAR START

Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxxx)

Bit	Reset	Description
31:12	0x0	MSI_AXI_BAR_START: The start of the upstream AXI address space for MSI BAR. The upstream FPCI address is compared to start/1KB range for the MSI BAR to determine if the access is MSI. Bits 31:12 of MSI BAR start correspond to AXI address bits 31:12.

19.13.10.15 XUSB_HOST_MSI_FPCI_BAR_ST_0

MSI FPCI BAR START

Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000xxxx)

Bit	Reset	Description
31:4	0x0	MSI_FPCI_BAR_START: The start of the upstream FPCI address space for MSI BAR. The upstream FPCI address is compared to start/1KB range for MSI BAR to determine if the access is MSI. Bits 31:4 of MSI BAR start correspond to UFPCI address bits 39:12.

19.13.10.16 XUSB_HOST_MSI_VEC0_0

MSI VECTORi, i in [0,7], RW

Offset: 0x100 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR0: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

19.13.10.17 XUSB_HOST_MSI_VEC1_0

Offset: 0x104 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR1: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

19.13.10.18 XUSB_HOST_MSI_VEC2_0

Offset: 0x108 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR2: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

19.13.10.19 XUSB_HOST_MSI_VEC3_0

Offset: 0x10c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR3: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

19.13.10.20 XUSB_HOST_MSI_VEC4_0

Offset: 0x110 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR4: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

19.13.10.21 XUSB_HOST_MSI_VEC5_0

Offset: 0x114 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR5: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

19.13.10.22 XUSB_HOST_MSI_VEC6_0

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

Bit	Reset	Description
31:0	0x0	MSI_VECTOR6: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

19.13.10.23 XUSB_HOST_MSI_VEC7_0

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

Bit	Reset	Description
31:0	0x0	MSI_VECTOR7: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

19.13.10.24 XUSB_HOST_MSI_EN_VEC0_0

MSI ENABLE VECTORi, i in [0,7], RW

Offset: 0x140 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR0: Each vector register corresponds to the enable bit for 32 of the possible 256

Bit	Reset	Description
		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.

19.13.10.25 XUSB_HOST_MSI_EN_VEC1_0

Offset: 0x144 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR1: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

19.13.10.26 XUSB_HOST_MSI_EN_VEC2_0

Offset: 0x148 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR2: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

19.13.10.27 XUSB_HOST_MSI_EN_VEC3_0

Offset: 0x14c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR3: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

19.13.10.28 XUSB_HOST_MSI_EN_VEC4_0

Offset: 0x150 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR4: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

19.13.10.29 XUSB_HOST_MSI_EN_VEC5_0

Offset: 0x154 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR5: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

19.13.10.30 XUSB_HOST_MSI_EN_VEC6_0

Offset: 0x158 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR6: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

19.13.10.31 XUSB_HOST_MSI_EN_VEC7_0

Offset: 0x15c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR7: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

19.13.10.32 XUSB_HOST_CONFIGURATION_0

Configuration

Offset: 0x180 | Read/Write: R/W | Reset: 0x800X8X40 (0b1xxxxxxxxxxx1xxx10xxxxxx01000000)

Bit	R/W	Reset	Description
31	RW	0x1	CLKEN_OVERRIDE: This can override the clock enable in case of a malfunction.
19	RW	0x1	PW_NO_DEVSEL_ERR_CYA: Setting this bit disables detection of DECERR due to no DEVSEL for DS PWs only. What are DS and PW? PW= pass write?
18	RO	X	INITIATOR_READ_IDLE: This read-only bit provides status reads on AFI upstream A value of 1b indicates there are no outstanding reads to the initiator.
17	RO	X	INITIATOR_WRITE_IDLE: This read-only bit provides status writes on AFI upstream A value of 1b indicates there are no outstanding writes to the initiator.
15	RW	0x1	WDATA_LEAD_CYA: Used to en(dis)able the handling of write data ahead of requests on IPFS AXI target.
14	RW	0x0	WR_INTRLV_CYA: Used to en(dis)able the handling of interleaved write requests on IPFS AXI target.
11	RO	X	TARGET_READ_IDLE: This read-only bit provides status reads to IPFS target. A value of 1b indicates there are no outstanding reads to the downstream FPCI.
10	RO	X	TARGET_WRITE_IDLE: This read-only bit provides status writes to IPFS target. A value of 1b indicates there are no outstanding writes to the downstream FPCI.
9	RO	X	MSI_VEC_EMPTY: This read-only bit provides status on whether MSI Vector registers have any active bits valid or not.
7	RW	0x0	UFPCI_MSIAW: MSI After Write ordering rule. 1 = Whenever MSI is ready assert the interrupt 0 = Default behavior, apply MSIAW ordering rule
6	RW	0x1	UFPCI_PWPASSPW: Input to the upstream FPCI 1 = Whenever write is ready, send it. 0 = Write goes only when outstanding PWs outside of new write's region are retired (default).
5	RW	0x0	UFPCI_PASSPW: Input to the upstream FPCI. Allows the upstream FPCI reads to pass writes.
4	RW	0x0	UFPCI_PWPASSNPW: Used for the upstream FPCI. Allows the upstream FPCI PWs to pass NPWs.
3	RW	0x0	DFPCI_PWPASSNPW: Used for the downstream FPCI. Allows the downstream FPCI PWs to pass NPWs.

Bit	R/W	Reset	Description
2	RW	0x0	DFPCI_RSPPASSPW: Input to the downstream FPCI. Allows the downstream FPCI responses to pass writes
1	RW	0x0	DFPCI_PASSPW: Input to the downstream FPCI. Allow the downstream FPCI reads to pass writes.
0	RW	0x0	EN_FPCI: When the IPFS device block is disabled, it is completely invisible on the IPFS bus; i.e., it does not even process IPFS configuration accesses.

19.13.10.33 XUSB_HOST_FPCI_ERROR_MASKS_0

FPCI Error Masks

Offset: 0x184 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	MASK_FPCI_MASTER_ABORT: This bit allows an FPCI error to be forwarded to an AXI response when the FPCI error response indicates a Master Abort. 1 = Forward error 0 = Return AXI OKAY response (2'b0)
1	0x0	MASK_FPCI_DATA_ERROR: This bit allows an FPCI error to be forwarded to an AXI response when the FPCI error response indicates a Data Error. 1 = Forward error 0 = Return AXI OKAY response (2'b0)
0	0x0	MASK_FPCI_TARGET_ABORT: This bit allows an FPCI error to be forwarded to an AXI response when FPCI error response indicates a Target Abort. This bit also covers a decode error generated when there is no DEVSEL received. 1 = Forward error 0 = Return AXI OKAY response (2'b0)

19.13.10.34 XUSB_HOST_INTR_MASK_0

Interrupt Masks

Offset: 0x188 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxxxxxx0xxxxxxx0)

Bit	Reset	Description
16	0x0	IP_INT_MASK: IP (SATA/AZA) interrupt to the CPU complex gated by the mask.
8	0x0	MSI_MASK: MSI to the CPU complex gated by the mask.
0	0x0	INT_MASK: Interrupt to the CPU complex gated by the mask.

19.13.10.35 XUSB_HOST_INTR_CODE_0

Interrupt Control

Offset: 0x18c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4:0	0x0	INT_CODE: Eight interrupt codes. If the code is 0, logging of the next interrupt is enabled 0 = INT_CODE_CLEAR : Clear interrupt code 1 = INT_CODE_INI_SLVERR: Interrupt code for CPU AXI SLVERR response to IPFS 2 = INT_CODE_INI_DECERR: Interrupt code for CPU AXI DECERR response to IPFS 3 = INT_CODE_TGT_SLVERR: Interrupt code for PCIe endpoint FPCI target abort or data error response to IPFS 4 = INT_CODE_TGT_DECERR: Interrupt code for PCIe2 FPCI master abort response to IPFS 5 = INT_CODE_TGT_WRERR: Interrupt code for bufferable write to non-posted write address region 6 = RSVD1: Reserved

Bit	Reset	Description
		7 = INT_CODE_DFPCI_DECERR: Interrupt code for PCIe2 response to downstream request when downstream FPCI address does not fall in a claimable downstream region 8 = INT_CODE_AXI_DECERR: Interrupt code for IPFS response to downstream request when AXI target AXI address does not fall in any IPFS downstream BARs 9 = INT_CODE_FPCI_TIMEOUT: Interrupt code for FPCI Timeout 10 = RSVD2: Reserved for future expansion 11 = RSVD3 12 = RSVD4 13 = RSVD5 14 = RSVD6 15 = INT_CODE_SM_FATAL_ERROR: Interrupt code for SM fatal error 16 = INT_CODE_SM_NON_FATAL_ERROR: Interrupt code for SM non-fatal error

19.13.10.36 XUSB_HOST_INTR_SIGNATURE_0

Interrupt Signature

Offset: 0x190 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000x0)

Bit	Reset	Description
31:2	0x0	INT_INFO: For interrupt codes 1-5/7-8, this field contains address bits [31:2], either in FPCI memory space or AXI space. For FPCI generated errors, the field contains the FPCI address. For AXI/IPFS generated errors, the field contains the AXI address.
0	0x0	DIR: Indicates the direction of the AXI/FPCI transaction. 1=RD/0=WRIF signature type is 6 (sideband message), this field is 1. 0 = WRITE: Interrupt due to a write transaction 1 = READ: Interrupt due to a read transaction

19.13.10.37 XUSB_HOST_UPPER_FPCI_ADDR_0

Upper FPCI Address

Offset: 0x194 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	INT_INFO_UPPER: These 8 bits are the upper byte of the captured FPCI address (bits [39:32]) when the interrupt code is 3, 4, or 7. These bits determine the region in the Hypertransport Address Map that was accessed.

19.13.10.38 XUSB_HOST_IPFS_INTR_ENABLE_0

IPFS Interrupt Enable

Offset: 0x198 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xxxx00000000)

Bit	Reset	Description
13	0x0	EN_SM_NON_FATAL_ERROR: Enable bit for interrupt code 15
12	0x0	EN_SM_FATAL_ERROR: Enable bit for interrupt code 14
7	0x0	EN_FPCI_TIMEOUT: Enable bit for interrupt code 9
6	0x0	EN_AXI_DECERR: Enable bit for interrupt code 8
5	0x0	EN_DFPCI_DECERR: Enable bit for interrupt code 7
4	0x0	EN_TGT_WRERR: Enable bit for interrupt code 5
3	0x0	EN_TGT_DECERR: Enable bit for interrupt code 4
2	0x0	EN_TGT_SLVERR: Enable bit for interrupt code 3

Bit	Reset	Description
1	0x0	EN_INI_DECERR: Enable bit for interrupt code 2
0	0x0	EN_INI_SLVERR: Enable bit for interrupt code 1

19.13.10.39 XUSB_HOST_UFPCI_CONFIG_0

Upstream FPCI Configuration

Offset: 0x19c | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00010)

Bit	Reset	Description
4:0	0x2	UNITID_T0C0: Upstream FPCI Unit ID for controller 0. HyperTransport, upstream FPCI request

19.13.10.40 XUSB_HOST_CFG_REVID_0

CFG_REVID Register

Offset: 0x1a0 | Read/Write: R/W | Reset: 0x000X10XX (0bxxxxxxxxxxxxxxxxxx0100xxxxxxx1xx)

Bit	R/W	Reset	Description
19	RO	X	DEV2SM_NONISO_REQUEST_PEND: This bit indicates if there is a non-ISO request pending. 0 = NO 1 = YES
18	RO	X	DEV2SM_ISO_REQUEST_PEND: This bit indicates if there is an ISO request pending. 0 = NO 1 = YES
13:12	RW	0x1	STRAP_CPU_MODE: MCP: Mode to send MSI. It can be programmable. 0 = NB_INTEL 1 = NB_AMD 2 = AMD 3 = TMTA
11	RW	0x0	CFG_REVID_WRITE_ENABLE: MCP: The enable to override the rev ID. It can be programmable. 0 = CLEAR 1 = SET
10	RW	0x0	CFG_REVID_OVERRIDE: MCP: Provides a way to override the current revision ID. It can be programmable. 0 = DISABLE 1 = ENABLE
4	RO	X	DEV2LEG_NONCOH_REQUEST_PEND: MCP: Tells the leg block that a non-coherent request is pending. 0 = NO 1 = YES
3	RO	X	DEV2LEG_COH_REQUEST_PEND: MCP comment: Tells the leg block that a coherent request is pending. 0 = NO 1 = YES
2	RW	0x1	SM2DEV_FPCI_TIMEOUT_EN: FPCI timeout enable bit for Controller 0 = DISABLE 1 = ENABLE

19.13.10.41 XUSB_HOST_FPCI_TIMEOUT_0

FPCI_TIMEOUT Register

Offset: 0x1a4 | Read/Write: R/W | Reset: 0x000f0000 (0bxxxxxxxxxxxxx11110000000000000000)

Bit	Reset	Description
19:0	0xf0000	SM2ALL_FPCI_TIMEOUT_THRESH: This field sets the timeout threshold value for the FPCI bus. It starts counting for each queue (ISO/NISO- RD/WR) with a pending request in the FPCI wrapper, the count resets when the requests are popped.

19.13.10.42 XUSB_HOST_TOM_0

Top of Memory Limit

Offset: 0x1a8 | Read/Write: R/W | Reset: 0x3fff0fff (0bxx11111111111111xxxx111111111111)

Bit	Reset	Description
29:16	0x3fff	LEG2ALL_TOM2: Top of Memory Limit 2.
11:0	0xfff	LEG2ALL_TOM1: Top of Memory Limit 1.

19.13.10.43 XUSB_HOST_INITIATOR_ISO_PW_RESP_PENDING_0

Initiator ISO PW Response Pending

Offset: 0x1ac | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	NUM_INITIATOR_ISO_PW_RESP_PEND: Number of pending initiator ISO PW responses

19.13.10.44 XUSB_HOST_INITIATOR_NISO_PW_RESP_PENDING_0

Initiator Non-ISO PW Response Pending

Offset: 0x1b0 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	NUM_INITIATOR_NISO_PW_RESP_PEND: Number of pending initiator NISO PW responses

19.13.10.45 XUSB_HOST_INTR_STATUS_0

IPFS Interrupt Status

Offset: 0x1b4 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2	X	IP_INTR_STATUS: Status of IP (SATA/AZA) interrupt
1	X	MSI_INTR_STATUS: Status of MSI interrupt
0	X	IPFS_INTR_STATUS: Status of IPFS interrupt

19.13.10.46 XUSB_HOST_DFPCI_BEN_0

Downstream FPCI Byte Enables

Offset: 0x1b8 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
31	0x0	EN_DFPCI_BEN: Enable bit for BEN. When set, the programmed BE is sent on the DFPCI bus

Bit	Reset	Description
3:0	0x0	DFPCI_BYTE_ENABLE_N: Active low byte enables

19.13.10.47 XUSB_HOST_CLKGATE_HYSTERESIS_0

Offset: 0x1bc | Read/Write: R/W | Reset: 0x00000014 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00010100)

Bit	Reset	Description
7:0	0x14	CLK_DISABLE_CNT: Number of IPFS clock cycles to wait after clock gating criteria are met to disable IPFS/FPCI clocks

19.13.10.48 XUSB_HOST_XUSB_HOST_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control Register

> **Note:** The FIFO timing aspects of this register are no longer supported, but are retained for software compatibility.

The clock override/ovr_mode fields of this register control the second-level clock gating for the client and MC side of the MCCIF. All clock gating is enabled by default.

A '1' written to the rclk/wclk field results in one of the following:

- With wclk/rclk override mode = LEGACY, the clock reverts to the legacy mode of operation (where the clock is on whenever the client clock is enabled).
- With wclk/rclk override mode = ON, the clock is always on inside the MCCIF and PC.

A '1' written to the cclk override field keeps the client's clock always on inside the MCCIF.Offset: 0x1dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000xxxxxxxxxxx0000)

Bit	Reset	Description
20	LEGACY	XUSB_HOST_RCLK_OVR_MODE: 0 = LEGACY 1 = ON
19	LEGACY	XUSB_HOST_WCLK_OVR_MODE: 0 = LEGACY 1 = ON
18	0x0	XUSB_HOST_CCLK_OVERRIDE
17	0x0	XUSB_HOST_RCLK_OVERRIDE
16	0x0	XUSB_HOST_WCLK_OVERRIDE
3	DISABLE	XUSB_HOST_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	XUSB_HOST_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	XUSB_HOST_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	XUSB_HOST_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

19.13.11 XUSB Host IFPS Registers

19.13.11.1 XUSB_HOST_ORDERING_RULES_0

Offset: 0x1e0 | Read/Write: R/W | Reset: 0x00000000 (0b000)

Bit	Reset	Description
3	0x0	UPSTREAM_MSIAW: Modified Upstream MSIAW ordering. 0 = Tegra K1 MSIAW behavior 1 = Legacy (Tegra 3) MSIAW behavior
2	0x0	UPSTREAM_RESPAW: Modified RespAW ordering. 0 = Tegra K1 RespAW behavior 1 = Legacy (Tegra 3) RespAW behavior
1	0x0	UPSTREAM_RAW: Modified RAW ordering. 0 = Tegra K1 RAW behavior 1 = Legacy (Tegra 3) RAW behavior

19.13.11.2 XUSB_HOST_A2F_UFPCI_CFG0_0

Offset: 0x1e4 | Read/Write: R/W | Reset: 0x00000050 (0b00000000xxxxx0000000000001010000)

Bit	Reset	Description
31:24	0x0	STATIC_WAIT_IDLE_CNTR
18:16	0x0	STATIC_UFPCI_UFA_STARVE_CNTR_PRI1
15:12	0x0	STATIC_UFPCI_UFA_STARVE_CNTR_PRI0
11:10	0x0	STATIC_UFPCI_RR_BURST_SZ_PRI1
9:8	0x0	STATIC_UFPCI_RR_BURST_SZ_PRI0
7	0x0	STATIC_WAIT_CLAMP_EN
6	0x1	STATIC_UFPCI_UFA_DYN_BLOCK_EN
5	0x0	STATIC_UFPCI_UFA_BLK_COHERENT
4:2	0x4	STATIC_UFPCI_BLOCK_CMD_THRESHOLD
1	0x0	STATIC_CYA_UFA_ARB
0	0x0	STATIC_CYA_BACK2BACK_UPSTREAM_BLOCK

19.13.11.3 XUSB_HOST_A2F_UFPCI_CFG1_0

Offset: 0x1e8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000)

Bit	Reset	Description
7:0	0x0	STATIC_WAIT_UNCLAMP_CNTR

19.13.12 XUSB PADCTL Registers

19.13.12.1 XUSB_PADCTL_BOOT_MEDIA_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4:1	0x0	BOOT_PORT: 0 = USB2_OTG0 1 = USB2_OTG1 2 = USB2_OTG2 3 = USB2_OTG3 4 = USB2_OTG4 5 = USB2_OTG5 6 = USB2_OTG6 7 = ULPI 9 = HSIC0 10 = HSIC1
0	0x0	BOOT_MEDIA_ENABLE: 0 = NO 1 = YES

19.13.12.2 XUSB_PADCTL_USB2_PAD_MUX_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00x0xxxxx000000)

Bit	Reset	Description
15	0x0	USB2_HSIC_PAD_PORT1: 0 = SNPS 1 = XUSB
14	0x0	USB2_HSIC_PAD_PORT0: 0 = SNPS 1 = XUSB
12	0x0	USB2_ULPI_PAD_PORT: 0 = SNPS 1 = XUSB
5:4	0x0	USB2_OTG_PAD_PORT2: 0 = SNPS 1 = XUSB 2 = UART
3:2	0x0	USB2_OTG_PAD_PORT1: 0 = SNPS 1 = XUSB 2 = UART
1:0	0x0	USB2_OTG_PAD_PORT0: 0 = SNPS 1 = XUSB 2 = UART

19.13.12.3 XUSB_PADCTL_USB2_PORT_CAP_0

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00xxxxxxxxxxx000000000000)

Bit	Reset	Description
25	0x0	ULPI_PORT_INTERNAL: 0 = NO 1 = YES
24	0x0	ULPI_PORT_CAP: 0 = ULPI_MASTER 1 = ULPI_PHY
11	0x0	PORT2_REVERSE_ID: 0 = NO

Bit	Reset	Description
		1 = YES
10	0x0	PORT2_INTERNAL: 0 = NO 1 = YES
9:8	0x0	PORT2_CAP: 0 = DISABLED 1 = HOST_ONLY 2 = DEVICE_ONLY 3 = OTG_CAP
7	0x0	PORT1_REVERSE_ID: 0 = NO 1 = YES
6	0x0	PORT1_INTERNAL: 0 = NO 1 = YES
5:4	0x0	PORT1_CAP: 0 = DISABLED 1 = HOST_ONLY 2 = DEVICE_ONLY 3 = OTG_CAP
3	0x0	PORT0_REVERSE_ID: 0 = NO 1 = YES
2	0x0	PORT0_INTERNAL: 0 = NO 1 = YES
1:0	0x0	PORT0_CAP: 0 = DISABLED 1 = HOST_ONLY 2 = DEVICE_ONLY 3 = OTG_CAP

19.13.12.4 XUSB_PADCTL_SNPS_OC_MAP_0

Offset: 0xc | Read/Write: R/W | Reset: 0x000001ff (0bxxxxxxxxxxxxxxxxxxxxxxxxx111111111)

Bit	Reset	Description
8:6	0x7	CONTROLLER3_OC_PIN: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 6 = OC_DETECTED_VBUS_PAD2 7 = OC_DETECTION_DISABLED
5:3	0x7	CONTROLLER2_OC_PIN: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 6 = OC_DETECTED_VBUS_PAD2 7 = OC_DETECTION_DISABLED
2:0	0x7	CONTROLLER1_OC_PIN:

Bit	Reset	Description
		0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 6 = OC_DETECTED_VBUS_PAD2 7 = OC_DETECTION_DISABLED

19.13.12.5 XUSB_PADCTL_USB2_OC_MAP_0

Offset: 0x10 | Read/Write: R/W | Reset: 0x000001ff (0bxxxxxxxxxxxxxxxxxxxxxxxxx111111111)

Bit	Reset	Description
8:6	0x7	PORT2_OC_PIN: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 6 = OC_DETECTED_VBUS_PAD2 7 = OC_DETECTION_DISABLED
5:3	0x7	PORT1_OC_PIN: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 6 = OC_DETECTED_VBUS_PAD2 7 = OC_DETECTION_DISABLED
2:0	0x7	PORT0_OC_PIN: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 6 = OC_DETECTED_VBUS_PAD2 7 = OC_DETECTION_DISABLED

19.13.12.6 XUSB_PADCTL_SS_PORT_MAP_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000077 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01110111)

Bit	Reset	Description
7	0x0	PORT1_INTERNAL: 0 = NO 1 = YES
6:4	0x7	PORT1_MAP: 0 = USB2_PORT0 1 = USB2_PORT1 2 = USB2_PORT2 7 = INIT_DISABLED
3	0x0	PORT0_INTERNAL: 0 = NO 1 = YES

Bit	Reset	Description
2:0	0x7	PORT0_MAP: 0 = USB2_PORT0 1 = USB2_PORT1 2 = USB2_PORT2 7 = INIT_DISABLED

19.13.12.7 XUSB_PADCTL_OC_DET_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x0000fce0 (0bx0000000x00000001111110011100000)

Bit	Reset	Description
30	0x0	OC_DETECTED_INTERRUPT_ENABLE_VBUSPAD2: 0 = NO 1 = YES
29	0x0	OC_DETECTED_INTERRUPT_ENABLE_VBUSPAD1: 0 = NO 1 = YES
28	0x0	OC_DETECTED_INTERRUPT_ENABLE_VBUSPAD0: 0 = NO 1 = YES
27	0x0	OC_DETECTED_INTERRUPT_ENABLE3: 0 = NO 1 = YES
26	0x0	OC_DETECTED_INTERRUPT_ENABLE2: 0 = NO 1 = YES
25	0x0	OC_DETECTED_INTERRUPT_ENABLE1: 0 = NO 1 = YES
24	0x0	OC_DETECTED_INTERRUPT_ENABLE0: 0 = NO 1 = YES
22	0x0	OC_DETECTED_VBUS_PAD2: 0 = NO 1 = YES
21	0x0	OC_DETECTED_VBUS_PAD1: 0 = NO 1 = YES
20	0x0	OC_DETECTED_VBUS_PAD0: 0 = NO 1 = YES
19	0x0	OC_DETECTED3: 0 = NO 1 = YES
18	0x0	OC_DETECTED2: 0 = NO 1 = YES
17	0x0	OC_DETECTED1: 0 = NO 1 = YES
16	0x0	OC_DETECTED0: 0 = NO 1 = YES
15:13	0x7	VBUS_ENABLE1_OC_MAP:

Bit	Reset	Description
		0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 6 = OC_DETECTED_VBUS_PAD2 7 = OC_DETECTION_DISABLED
12:10	0x7	VBUS_ENABLE0_OC_MAP: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 6 = OC_DETECTED_VBUS_PAD2 7 = OC_DETECTION_DISABLED
9	0x0	VBUS_ENABLE1: 0 = NO 1 = YES
8	0x0	VBUS_ENABLE0: 0 = NO 1 = YES
7:5	0x7	VBUS_ENABLE2_OC_MAP: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 6 = OC_DETECTED_VBUS_PAD2 7 = OC_DETECTION_DISABLED
4	0x0	VBUS_ENABLE2: 0 = NO 1 = YES
3	0x0	SET_OC_DETECTED3: 0 = NO 1 = YES
2	0x0	SET_OC_DETECTED2: 0 = NO 1 = YES
1	0x0	SET_OC_DETECTED1: 0 = NO 1 = YES
0	0x0	SET_OC_DETECTED0: 0 = NO 1 = YES

19.13.12.8 XUSB_PADCTL_ELPG_PROGRAM_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x07770000 (0bxxxxx111x111x11100x0000000x00000)

Bit	Reset	Description
26	0x1	AUX_MUX_LP0_VCORE_DOWN: 0 = NO 1 = YES
25	0x1	AUX_MUX_LP0_CLAMP_EN_EARLY: 0 = NO

Bit	Reset	Description
		1 = YES
24	0x1	AUX_MUX_LP0_CLAMP_EN: 0 = NO 1 = YES
22	0x1	SSP1_ELPG_VCORE_DOWN: 0 = NO 1 = YES
21	0x1	SSP1_ELPG_CLAMP_EN_EARLY: 0 = NO 1 = YES
20	0x1	SSP1_ELPG_CLAMP_EN: 0 = NO 1 = YES
18	0x1	SSP0_ELPG_VCORE_DOWN: 0 = NO 1 = YES
17	0x1	SSP0_ELPG_CLAMP_EN_EARLY: 0 = NO 1 = YES
16	0x1	SSP0_ELPG_CLAMP_EN: 0 = NO 1 = YES
15	0x0	SS_PORT1_WAKEUP_EVENT: 0 = NO 1 = YES
14	0x0	SS_PORT0_WAKEUP_EVENT: 0 = NO 1 = YES
12	0x0	USB2_HSIC_PORT1_WAKEUP_EVENT: 0 = NO 1 = YES
11	0x0	USB2_HSIC_PORT0_WAKEUP_EVENT: 0 = NO 1 = YES
10	0x0	USB2_PORT2_WAKEUP_EVENT: 0 = NO 1 = YES
9	0x0	USB2_PORT1_WAKEUP_EVENT: 0 = NO 1 = YES
8	0x0	USB2_PORT0_WAKEUP_EVENT: 0 = NO 1 = YES
7	0x0	SS_PORT1_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES
6	0x0	SS_PORT0_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES
4	0x0	USB2_HSIC_PORT1_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES
3	0x0	USB2_HSIC_PORT0_WAKE_INTERRUPT_ENABLE: 0 = NO

Bit	Reset	Description
		1 = YES
2	0x0	USB2_PORT2_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES
1	0x0	USB2_PORT1_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES
0	0x0	USB2_PORT0_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES

19.13.12.9 XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL0_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x00XX00XX (0b000000000x000x000000000000x000x1)

Bit	R/W	Reset	Description
31	RW	0x0	GENERATE_SRP: 0 = NO 1 = YES
30	RW	0x0	SRP_INTR_EN: 0 = NO 1 = YES
29	RW	0x0	SRP_DETECTED: 0 = NO 1 = YES
28	RW	0x0	SRP_DETECT_EN: 0 = NO 1 = YES
27	RW	0x0	DCD_INTR_EN: 0 = NO 1 = YES
26	RW	0x0	DCD_DETECTED: 0 = NO 1 = YES
25	RW	0x0	ZIN_FILTER_EN: 0 = NO 1 = YES
24	RW	0x0	ZIN_CHNG_INTR_EN: 0 = NO 1 = YES
23	RW	0x0	ZIN_ST_CHNG: 0 = NO 1 = YES
22	RO	X	ZIN: 0 = NO 1 = YES
21	RW	0x0	ZIP_FILTER_EN: 0 = NO 1 = YES
20	RW	0x0	ZIP_CHNG_INTR_EN: 0 = NO 1 = YES
19	RW	0x0	ZIP_ST_CHNG: 0 = NO 1 = YES

Bit	R/W	Reset	Description
18	RO	X	ZIP: 0 = NO 1 = YES
17	RW	0x0	USBON_RPU: 0 = NO 1 = YES
16	RW	0x0	USBON_RPD: 0 = NO 1 = YES
15	RW	0x0	USBOP_RPU: 0 = NO 1 = YES
14	RW	0x0	USBOP_RPD: 0 = NO 1 = YES
13	RW	0x0	OP_I_SRC_EN: 0 = NO 1 = YES
12	RW	0x0	ON_SRC_EN: 0 = NO 1 = YES
11	RW	0x0	ON_SINK_EN: 0 = NO 1 = YES
10	RW	0x0	OP_SRC_EN: 0 = NO 1 = YES
9	RW	0x0	OP_SINK_EN: 0 = NO 1 = YES
8	RW	0x0	VDAT_DET_FILTER_EN: 0 = NO 1 = YES
7	RW	0x0	VDAT_DET_CHNG_INTR_EN: 0 = NO 1 = YES
6	RW	0x0	VDAT_DET_ST_CHNG: 0 = NO 1 = YES
5	RO	X	VDAT_DET: 0 = NO 1 = YES
4	RW	0x0	VDCD_DET_FILTER_EN: 0 = NO 1 = YES
3	RW	0x0	VDCD_DET_CHNG_INTR_EN: 0 = NO 1 = YES
2	RW	0x0	VDCD_DET_ST_CHNG: 0 = NO 1 = YES
1	RO	X	VDCD_DET: 0 = NO 1 = YES

Bit	R/W	Reset	Description
0	RW	0x1	PD_CHG: 0 = NO 1 = YES

19.13.12.10 XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL1_0

Offset: 0x24 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0xxxx)

Bit	R/W	Reset	Description
4	RW	0x0	DIV_DET_EN: 0 = NO 1 = YES
3	RO	X	VOP_DIV2P7_DET: 0 = NO 1 = YES
2	RO	X	VOP_DIV2P0_DET: 0 = NO 1 = YES
1	RO	X	VON_DIV2P7_DET: 0 = NO 1 = YES
0	RO	X	VON_DIV2P0_DET: 0 = NO 1 = YES

19.13.12.11 XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD1_CTL0_0

Offset: 0x28 | Read/Write: R/W | Reset: 0x00XX00XX (0b000000000x000x000000000000x000x1)

Bit	R/W	Reset	Description
31	RW	0x0	GENERATE_SRP: 0 = NO 1 = YES
30	RW	0x0	SRP_INTR_EN: 0 = NO 1 = YES
29	RW	0x0	SRP_DETECTED: 0 = NO 1 = YES
28	RW	0x0	SRP_DETECT_EN: 0 = NO 1 = YES
27	RW	0x0	DCD_INTR_EN: 0 = NO 1 = YES
26	RW	0x0	DCD_DETECTED: 0 = NO 1 = YES
25	RW	0x0	ZIN_FILTER_EN: 0 = NO 1 = YES
24	RW	0x0	ZIN_CHNG_INTR_EN: 0 = NO 1 = YES
23	RW	0x0	ZIN_ST_CHNG:

Bit	R/W	Reset	Description
			0 = NO 1 = YES
22	RO	X	ZIN: 0 = NO 1 = YES
21	RW	0x0	ZIP_FILTER_EN: 0 = NO 1 = YES
20	RW	0x0	ZIP_CHNG_INTR_EN: 0 = NO 1 = YES
19	RW	0x0	ZIP_ST_CHNG: 0 = NO 1 = YES
18	RO	X	ZIP: 0 = NO 1 = YES
17	RW	0x0	USBON_RPU: 0 = NO 1 = YES
16	RW	0x0	USBON_RPD: 0 = NO 1 = YES
15	RW	0x0	USBOP_RPU: 0 = NO 1 = YES
14	RW	0x0	USBOP_RPD: 0 = NO 1 = YES
13	RW	0x0	OP_I_SRC_EN: 0 = NO 1 = YES
12	RW	0x0	ON_SRC_EN: 0 = NO 1 = YES
11	RW	0x0	ON_SINK_EN: 0 = NO 1 = YES
10	RW	0x0	OP_SRC_EN: 0 = NO 1 = YES
9	RW	0x0	OP_SINK_EN: 0 = NO 1 = YES
8	RW	0x0	VDAT_DET_FILTER_EN: 0 = NO 1 = YES
7	RW	0x0	VDAT_DET_CHNG_INTR_EN: 0 = NO 1 = YES
6	RW	0x0	VDAT_DET_ST_CHNG: 0 = NO 1 = YES
5	RO	X	VDAT_DET: 0 = NO

Bit	R/W	Reset	Description
			1 = YES
4	RW	0x0	VDCD_DET_FILTER_EN: 0 = NO 1 = YES
3	RW	0x0	VDCD_DET_CHNG_INTR_EN: 0 = NO 1 = YES
2	RW	0x0	VDCD_DET_ST_CHNG: 0 = NO 1 = YES
1	RO	X	VDCD_DET: 0 = NO 1 = YES
0	RW	0x1	PD_CHG: 0 = NO 1 = YES

19.13.12.12 XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD1_CTL1_0

Offset: 0x2c | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0xxxx)

Bit	R/W	Reset	Description
4	RW	0x0	DIV_DET_EN: 0 = NO 1 = YES
3	RO	X	VOP_DIV2P7_DET: 0 = NO 1 = YES
2	RO	X	VOP_DIV2P0_DET: 0 = NO 1 = YES
1	RO	X	VON_DIV2P7_DET: 0 = NO 1 = YES
0	RO	X	VON_DIV2P0_DET: 0 = NO 1 = YES

19.13.12.13 XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD2_CTL0_0

Offset: 0x30 | Read/Write: R/W | Reset: 0x00XX00XX (0b000000000x000x000000000000x000x1)

Bit	R/W	Reset	Description
31	RW	0x0	GENERATE_SRP: 0 = NO 1 = YES
30	RW	0x0	SRP_INTR_EN: 0 = NO 1 = YES
29	RW	0x0	SRP_DETECTED: 0 = NO 1 = YES
28	RW	0x0	SRP_DETECT_EN: 0 = NO 1 = YES

Bit	R/W	Reset	Description
27	RW	0x0	DCD_INTR_EN: 0 = NO 1 = YES
26	RW	0x0	DCD_DETECTED: 0 = NO 1 = YES
25	RW	0x0	ZIN_FILTER_EN: 0 = NO 1 = YES
24	RW	0x0	ZIN_CHNG_INTR_EN: 0 = NO 1 = YES
23	RW	0x0	ZIN_ST_CHNG: 0 = NO 1 = YES
22	RO	X	ZIN: 0 = NO 1 = YES
21	RW	0x0	ZIP_FILTER_EN: 0 = NO 1 = YES
20	RW	0x0	ZIP_CHNG_INTR_EN: 0 = NO 1 = YES
19	RW	0x0	ZIP_ST_CHNG: 0 = NO 1 = YES
18	RO	X	ZIP: 0 = NO 1 = YES
17	RW	0x0	USBON_RPU: 0 = NO 1 = YES
16	RW	0x0	USBON_RPD: 0 = NO 1 = YES
15	RW	0x0	USBOP_RPU: 0 = NO 1 = YES
14	RW	0x0	USBOP_RPD: 0 = NO 1 = YES
13	RW	0x0	OP_I_SRC_EN: 0 = NO 1 = YES
12	RW	0x0	ON_SRC_EN: 0 = NO 1 = YES
11	RW	0x0	ON_SINK_EN: 0 = NO 1 = YES
10	RW	0x0	OP_SRC_EN: 0 = NO 1 = YES

Bit	R/W	Reset	Description
9	RW	0x0	OP_SINK_EN: 0 = NO 1 = YES
8	RW	0x0	VDAT_DET_FILTER_EN: 0 = NO 1 = YES
7	RW	0x0	VDAT_DET_CHNG_INTR_EN: 0 = NO 1 = YES
6	RW	0x0	VDAT_DET_ST_CHNG: 0 = NO 1 = YES
5	RO	X	VDAT_DET: 0 = NO 1 = YES
4	RW	0x0	VDCD_DET_FILTER_EN: 0 = NO 1 = YES
3	RW	0x0	VDCD_DET_CHNG_INTR_EN: 0 = NO 1 = YES
2	RW	0x0	VDCD_DET_ST_CHNG: 0 = NO 1 = YES
1	RO	X	VDCD_DET: 0 = NO 1 = YES
0	RW	0x1	PD_CHG: 0 = NO 1 = YES

19.13.12.14 PADCTL_USB2_BATTERY_CHRG_OTGPAD2_CTL1_0

Offset: 0x34 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0xxxx)

Bit	R/W	Reset	Description
4	RW	0x0	DIV_DET_EN: 0 = NO 1 = YES
3	RO	X	VOP_DIV2P7_DET: 0 = NO 1 = YES
2	RO	X	VOP_DIV2P0_DET: 0 = NO 1 = YES
1	RO	X	VON_DIV2P7_DET: 0 = NO 1 = YES
0	RO	X	VON_DIV2P0_DET: 0 = NO 1 = YES

19.13.12.15 XUSB_PADCTL_USB2_BATTERY_CHRG_BIASPAD_0

Offset: 0x38 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxx00000000xxxxxx00x00x0)

Bit	R/W	Reset	Description
20	RW	0x0	ID_OVERRIDE: 0 = NO 1 = YES
19:18	RW	0x0	ID_SOURCE_SELECT: 0 = NO_OVERRIDE 1 = GPIO 2 = VBUS_OVERRIDE
17	RW	0x0	VBUS_OVERRIDE: 0 = NO 1 = YES
16:15	RW	0x0	VBUS_SOURCE_SELECT: 0 = NO_OVERRIDE 1 = GPIO 2 = VBUS_OVERRIDE
14	RW	0x0	ID_CONNECT_CHNG_INTR_EN: 0 = NO 1 = YES
13	RW	0x0	ID_CONNECT_ST_CHNG: 0 = NO 1 = YES
12	RO	X	ID_CONNECT_STATUS: 0 = NO 1 = YES
11	RO	X	IDDIG_C: 0 = NO 1 = YES
10	RO	X	IDDIG_B: 0 = NO 1 = YES
9	RO	X	IDDIG_A: 0 = NO 1 = YES
8	RO	X	IDDIG: 0 = NO 1 = YES
6	RW	0x0	VBUS_VLD_CHNG_INTR_EN: 0 = NO 1 = YES
5	RW	0x0	VBUS_VLD_ST_CHNG: 0 = NO 1 = YES
4	RO	X	VBUS_VLD: 0 = NO 1 = YES
3	RW	0x0	OTG_VBUS_SESS_VLD_CHNG_INTR_EN: 0 = NO 1 = YES
2	RW	0x0	OTG_VBUS_SESS_VLD_ST_CHNG: 0 = NO 1 = YES
1	RO	X	OTG_VBUS_SESS_VLD: 0 = NO 1 = YES
0	RW	0x0	PD_OTG: 0 = NO

Bit	R/W	Reset	Description
			1 = YES

19.13.12.16 XUSB_PADCTL_USB2_BATTERY_CHRG_TDCD_DBNC_TIMER_0

Offset: 0x3c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx00000000000)

Bit	Reset	Description
10:0	0x0	TDCD_DBNC

19.13.12.17 XUSB_PADCTL_IOPHY_PLL_P0_CTL1_0

USB3/PCIE PLL0 control signals: One set per PLL

Offset: 0x40 | Read/Write: R/W | Reset: 0x3X2X0000 (0bxx11xxx0xx10xxx000000x0000000000)

Bit	R/W	Reset	Description
29:28	RW	0x3	PLL1_REFCLK_NDIV
27	RO	X	PLL1_LOCKDET
24	RW	0x0	PLL1_MODE
21:20	RW	0x2	PLL0_REFCLK_NDIV
19	RO	X	PLL0_LOCKDET
16	RW	0x0	PLL0_MODE
15:12	RW	0x0	REFCLK_SEL
11	RW	0x0	REFCLK_TERM100
9	RW	0x0	PLL_CKBUFPD_OVR
8	RW	0x0	PLL_CKBUFPD_M
7	RW	0x0	PLL_CKBUFPD_BL
6	RW	0x0	PLL_CKBUFPD_BR
5	RW	0x0	PLL_CKBUFPD_TL
4	RW	0x0	PLL_CKBUFPD_TR
3	RW	0x0	PLL_PWR_OVRD
2	RW	0x0	PLL_EMULATION_RST_
1	RW	0x0	PLL_RST_
0	RW	0x0	PLL_IDDQ

19.13.12.18 XUSB_PADCTL_IOPHY_PLL_P0_CTL2_0

Offset: 0x44 | Read/Write: R/W | Reset: 0xXX880037 (0bxxxxxxxx100010000x00000000110111)

Bit	R/W	Reset	Description
31:24	RO	X	PLL_MISC_OUT
23:20	RW	0x8	PLL1_CP_CNTL
19:16	RW	0x8	PLL0_CP_CNTL
15	RW	0x0	PLL_BYPASS_EN
13	RW	0x0	PLL_EMULATION_ON

Bit	R/W	Reset	Description
12	RW	0x0	TCLKOUT_EN
11:8	RW	0x0	TCLKOUT_SEL
7	RW	0x0	XDIGCLK4P5_EN
6	RW	0x0	REFCLKBUF_EN
5	RW	0x1	TXCLKREF_EN
4	RW	0x1	TXCLKREF_SEL
3	RW	0x0	XDIGCLK_EN
2:0	RW	0x7	XDIGCLK_SEL

19.13.12.19 XUSB_PADCTL_IOPHY_PLL_P0_CTL3_0

Offset: 0x48 | Read/Write: R/W | Reset: 0x0000XX0e (0b0000xx000000xx00x0xxxxxx0xx01110)

Bit	R/W	Reset	Description
31:28	RW	0x0	PLL_TEMP_CNTL
25:20	RW	0x0	PLL_BW_CNTL
17:16	RW	0x0	PLL_BGAP_CNTL
15	RO	X	RCAL_DONE
14	RW	0x0	RCAL_RESET
12:8	RO	X	RCAL_VAL
7	RW	0x0	RCAL_BYPASS
4:0	RW	0xe	RCAL_CODE

19.13.12.20 XUSB_PADCTL_IOPHY_PLL_P0_CTL4_0

Offset: 0x4c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PLL_MISC_CNTL

19.13.12.21 XUSB_PADCTL_IOPHY_USB3_PAD0_CTL_1_0

USB3 PADS: XUSB Static controls: Unit specific

Offset: 0x50 | Read/Write: R/W | Reset: 0x0000ec14 (0bxxxxxxxxxx00000001110110000010100)

Bit	Reset	Description
22:21	0x0	RX_DIV
20:19	0x0	TX_DIV
18:15	0x1	TX_DRV_CNTL
14:11	0xd	TX_CMADJ
10:5	0x20	TX_AMP
4:3	0x2	RX_RATE
2:1	0x2	TX_RATE
0	0x0	RATE_MODE

19.13.12.22 XUSB_PADCTL_IOPHY_USB3_PAD1_CTL_1_0

Offset: 0x54 | Read/Write: R/W | Reset: 0x0000ec14 (0bxxxxxxxxx00000011110110000010100)

Bit	Reset	Description
22:21	0x0	RX_DIV
20:19	0x0	TX_DIV
18:15	0x1	TX_DRV_CNTL
14:11	0xd	TX_CMADJ
10:5	0x20	TX_AMP
4:3	0x2	RX_RATE
2:1	0x2	TX_RATE
0	0x0	RATE_MODE

19.13.12.23 XUSB_PADCTL_IOPHY_USB3_PAD0_CTL_2_0

Offset: 0x58 | Read/Write: R/W | Reset: 0x24001000 (0b00100100000000000001000000000000)

Bit	Reset	Description
31:24	0x24	CDR_CNTL
23:8	0x10	RX_EQ
7:4	0x0	RX_WANDER
3:2	0x0	RX_TERM_CNTL
1:0	0x0	TX_TERM_CNTL

19.13.12.24 XUSB_PADCTL_IOPHY_USB3_PAD1_CTL_2_0

Offset: 0x5c | Read/Write: R/W | Reset: 0x24001000 (0b00100100000000000001000000000000)

Bit	Reset	Description
31:24	0x24	CDR_CNTL
23:8	0x10	RX_EQ
7:4	0x0	RX_WANDER
3:2	0x0	RX_TERM_CNTL
1:0	0x0	TX_TERM_CNTL

19.13.12.25 XUSB_PADCTL_IOPHY_USB3_PAD0_CTL_3_0

Offset: 0x60 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	EOM_CNTL

19.13.12.26 XUSB_PADCTL_IOPHY_USB3_PAD1_CTL_3_0

Offset: 0x64 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	EOM_CNTL

19.13.12.27 XUSB_PADCTL_IOPHY_USB3_PAD0_CTL_4_0

Offset: 0x68 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DFE_CNTL

19.13.12.28 XUSB_PADCTL_IOPHY_USB3_PAD1_CTL_4_0

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

Bit	Reset	Description
31:0	0x0	DFE_CNTL

19.13.12.29 XUSB_PADCTL_IOPHY_MISC_PAD_P0_CTL_1_0

USB3 PADS: Miscellaneous Dir Control: Common for all USB3/PCIE/SATA

USB3 PADS: Miscellaneous Ovrd: Common for all USB3/PCIE/SATA

Offset: 0x70 | Read/Write: R/W | Reset: 0x000aX330 (0bxxxx0000000010100xxx001100110000)

Bit	R/W	Reset	Description
27	RW	0x0	RX_PWR_OVRD
26	RW	0x0	TX_PWR_OVRD
25	RW	0x0	RATE_MODE_OVRD
24	RW	0x0	RATE_MODE
23:22	RW	0x0	RX_DIV
21:20	RW	0x0	TX_DIV
19:18	RW	0x2	RX_RATE
17:16	RW	0x2	TX_RATE
15	RW	0x0	TX_RDET
13	RO	X	TX_STAT_PRESENT
12	RO	X	RX_STAT_IDLE
11	RW	0x0	RX_DATA_EN
10	RW	0x0	RX_DATA_READY
9:8	RW	0x3	RX_SLEEP
7	RW	0x0	TX_DATA_EN
6	RW	0x0	TX_DATA_READY
5:4	RW	0x3	TX_SLEEP
3	RW	0x0	CKBUFPD_OVRD
2	RW	0x0	CKBUFPD
1	RW	0x0	IDDQ_OVRD
0	RW	0x0	IDDQ

19.13.12.30 XUSB_PADCTL_IOPHY_MISC_PAD_P1_CTL_1_0

Offset: 0x74 | Read/Write: R/W | Reset: 0x000aX330 (0bxxxx0000000010100xxx001100110000)

Bit	R/W	Reset	Description
27	RW	0x0	RX_PWR_OVRD
26	RW	0x0	TX_PWR_OVRD
25	RW	0x0	RATE_MODE_OVRD
24	RW	0x0	RATE_MODE
23:22	RW	0x0	RX_DIV
21:20	RW	0x0	TX_DIV
19:18	RW	0x2	RX_RATE
17:16	RW	0x2	TX_RATE
15	RW	0x0	TX_RDET
13	RO	X	TX_STAT_PRESENT
12	RO	X	RX_STAT_IDLE
11	RW	0x0	RX_DATA_EN
10	RW	0x0	RX_DATA_READY
9:8	RW	0x3	RX_SLEEP
7	RW	0x0	TX_DATA_EN
6	RW	0x0	TX_DATA_READY
5:4	RW	0x3	TX_SLEEP
3	RW	0x0	CKBUFPD_OVRD
2	RW	0x0	CKBUFPD
1	RW	0x0	IDDQ_OVRD
0	RW	0x0	IDDQ

19.13.12.31 XUSB_PADCTL_IOPHY_MISC_PAD_P0_CTL_2_0

Offset: 0x78 | Read/Write: R/W | Reset: 0xX0000000 (0bxx000x00xxxxxxxxx00000000000000)

Bit	R/W	Reset	Description
31:30	RO	X	SPARE_OUT
29:28	RW	0x0	SPARE_IN
27	RW	0x0	TEST_EN
25	RW	0x0	PRBS_CHK_EN
24	RW	0x0	PRBS_ERROR
13	RW	0x0	RX_CDR_RESET
12	RW	0x0	TX_SYNC
11	RW	0x0	FED_LOOP
10:8	RW	0x0	TX_DATA_MODE
7	RW	0x0	FEA_LOOP
6:4	RW	0x0	FEA_MODE
3	RW	0x0	NEA_LOOP
2	RW	0x0	NED_LOOP

Bit	R/W	Reset	Description
1:0	RW	0x0	NED_MODE

19.13.12.32 XUSB_PADCTL_IOPHY_MISC_PAD_P1_CTL_2_0

Offset: 0x7c | Read/Write: R/W | Reset: 0xX0000000 (0bxx000x00xxxxxxxxxx00000000000000)

Bit	R/W	Reset	Description
31:30	RO	X	SPARE_OUT
29:28	RW	0x0	SPARE_IN
27	RW	0x0	TEST_EN
25	RW	0x0	PRBS_CHK_EN
24	RW	0x0	PRBS_ERROR
13	RW	0x0	RX_CDR_RESET
12	RW	0x0	TX_SYNC
11	RW	0x0	FED_LOOP
10:8	RW	0x0	TX_DATA_MODE
7	RW	0x0	FEA_LOOP
6:4	RW	0x0	FEA_MODE
3	RW	0x0	NEA_LOOP
2	RW	0x0	NED_LOOP
1:0	RW	0x0	NED_MODE

19.13.12.33 XUSB_PADCTL_IOPHY_MISC_PAD_P0_CTL_3_0

Offset: 0x80 | Read/Write: R/W | Reset: 0x00040200 (0b00000000000001000000010xxxx0000)

Bit	Reset	Description
31:20	0x0	CDR_TEST
19	0x0	RX_IDLE_MODE_OVRD
18	0x1	RX_IDLE_MODE
17	0x0	RX_IDLE_BYP
16	0x0	TX_RDET_BYP
15:14	0x0	RX_IDLE_T
13:12	0x0	TX_RDET_T
11:8	0x2	TX_SEL_LOAD
3:0	0x0	MISC_CNTL

19.13.12.34 XUSB_PADCTL_IOPHY_MISC_PAD_P1_CTL_3_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00040200 (0b00000000000001000000010xxxx0000)

Bit	Reset	Description
31:20	0x0	CDR_TEST
19	0x0	RX_IDLE_MODE_OVRD

Bit	Reset	Description
18	0x1	RX_IDLE_MODE
17	0x0	RX_IDLE_BYP
16	0x0	TX_RDET_BYP
15:14	0x0	RX_IDLE_T
13:12	0x0	TX_RDET_T
11:8	0x2	TX_SEL_LOAD
3:0	0x0	MISC_CNTL

19.13.12.35 XUSB_PADCTL_IOPHY_MISC_PAD_P0_CTL_4_0

Offset: 0x88 | Read/Write: R/W | Reset: 0xXX000XX0 (0bx100x0000000xxxxxx0000x000x0xxxx)

Bit	R/W	Reset	Description
31	RO	X	AUX_RX_STAT_IDLE
30	RW	0x1	AUX_RX_IDLE_MODE
29	RW	0x0	AUX_RX_IDLE_EN
28	RW	0x0	AUX_RX_TERM_EN
27	RO	X	AUX_TX_STAT_PRESENT
26	RW	0x0	AUX_TX_RDET_CLK_EN
25	RW	0x0	AUX_TX_RDET_EN
24	RW	0x0	AUX_TX_TERM_EN
23	RW	0x0	AUX_MODE_OVRD
22	RW	0x0	AUX_HOLD_EN
21	RW	0x0	AUX_IDDQ_OVRD
20	RW	0x0	AUX_IDDQ
13	RW	0x0	TX_BYP_OVRD
12	RW	0x0	RX_BYP_MODE
11	RW	0x0	RX_BYP_EN
10	RW	0x0	RX_BYP_DIR
9	RO	X	RX_BYP_IN
8	RW	0x0	RX_BYP_OUT
7	RW	0x0	TX_BYP_EN
6	RW	0x0	TX_BYP_DIR
5	RO	X	TX_BYP_IN
4	RW	0x0	TX_BYP_OUT

19.13.12.36 XUSB_PADCTL_IOPHY_MISC_PAD_P1_CTL_4_0

Offset: 0x8c | Read/Write: R/W | Reset: 0xXX000XX0 (0bx100x0000000xxxxxx00000x00x0xxxx)

Bit	R/W	Reset	Description
31	RO	X	AUX_RX_STAT_IDLE

Bit	R/W	Reset	Description
30	RW	0x1	AUX_RX_IDLE_MODE
29	RW	0x0	AUX_RX_IDLE_EN
28	RW	0x0	AUX_RX_TERM_EN
27	RO	X	AUX_TX_STAT_PRESENT
26	RW	0x0	AUX_TX_RDET_CLK_EN
25	RW	0x0	AUX_TX_RDET_EN
24	RW	0x0	AUX_TX_TERM_EN
23	RW	0x0	AUX_MODE_OVRD
22	RW	0x0	AUX_HOLD_EN
21	RW	0x0	AUX_IDDQ_OVRD
20	RW	0x0	AUX_IDDQ
13	RW	0x0	TX_BYP_OVRD
12	RW	0x0	RX_BYP_MODE
11	RW	0x0	RX_BYP_EN
10	RW	0x0	RX_BYP_DIR
9	RW	0x0	RX_BYP_OUT
8	RO	X	RX_BYP_IN
7	RW	0x0	TX_BYP_EN
6	RW	0x0	TX_BYP_DIR
5	RO	X	TX_BYP_IN
4	RW	0x0	TX_BYP_OUT

19.13.12.37 XUSB_PADCTL_IOPHY_MISC_PAD_P0_CTL_5_0

Offset: 0x90 | Read/Write: R/W | Reset: 0x0000X0XX (0bxxxxxxxxxxxxxxxxxxxxxxxx00xx00xx0)

Bit	R/W	Reset	Description
17:12	RO	X	RX_QEYE_OUT
8	RW	0x0	RX_QEYE_EN
7	RW	0x0	EOM_EN
5	RO	X	EOM_TRAIN_DONE
4	RW	0x0	EOM_TRAIN_EN
3	RW	0x0	DFE_RESET
1	RO	X	DFE_TRAIN_DONE
0	RW	0x0	DFE_TRAIN_EN

19.13.12.38 XUSB_PADCTL_IOPHY_MISC_PAD_P1_CTL_5_0

Offset: 0x94 | Read/Write: R/W | Reset: 0x0000X0XX (0bxxxxxxxxxxxxxxxxxxxxxxxx00xx00xx0)

Bit	R/W	Reset	Description
17:12	RO	X	RX_QEYE_OUT

Bit	R/W	Reset	Description
8	RW	0x0	RX_QEYE_EN
7	RW	0x0	EOM_EN
5	RO	X	EOM_TRAIN_DONE
4	RW	0x0	EOM_TRAIN_EN
3	RW	0x0	DFE_RESET
1	RO	X	DFE_TRAIN_DONE
0	RW	0x0	DFE_TRAIN_EN

19.13.12.39 XUSB_PADCTL_IOPHY_MISC_PAD_P0_CTL_6_0

Offset: 0x98 | Read/Write: R/W | Reset: 0xXX000000 (0bxxxxxxxx00000000000000000000000)

Bit	R/W	Reset	Description
31:24	RO	X	MISC_OUT
23:16	RW	0x0	MISC_OUT_SEL
15:0	RW	0x0	MISC_TEST

19.13.12.40 XUSB_PADCTL_IOPHY_MISC_PAD_P1_CTL_6_0

Offset: 0x9c | Read/Write: R/W | Reset: 0xXX000000 (0bxxxxxxxx00000000000000000000000)

Bit	R/W	Reset	Description
31:24	RO	X	MISC_OUT
23:16	RW	0x0	MISC_OUT_SEL
15:0	RW	0x0	MISC_TEST

19.13.12.41 XUSB_PADCTL_USB2_OTG_PAD0_CTL_0_0

USB2 OTG, HSIC, ICUSB Control Settings

OTGPAD0, CTL0 static settings

One set per pad (except the BIAS pad which is just one set)

Offset: 0xa0 | Read/Write: R/W | Reset: 0x003808a0 (0bxxxxxxxx00111000000000100010100000)

Bit	Reset	Description
23	0x0	LSBIAS_SEL
22	0x0	DISCON_DETECT_METHOD
21	0x1	PD_ZI
20	0x1	PD2
19	0x1	PD
18	0x0	TERM_EN
17:16	0x0	LS_FSLEW
15:14	0x0	LS_RSLEW
13:12	0x0	FS_SLEW

Bit	Reset	Description
11:6	0x22	HS_SLEW
5:0	0x20	HS_CURR_LEVEL

19.13.12.42 XUSB_PADCTL_USB2_OTG_PAD1_CTL_0_0

OTGPAD1, CTL0 static settings

Offset: 0xa4 | Read/Write: R/W | Reset: 0x003808a0 (0bxxxxxxxx00111000000100010100000)

Bit	Reset	Description
23	0x0	LSBIAS_SEL
22	0x0	DISCON_DETECT_METHOD
21	0x1	PD_ZI
20	0x1	PD2
19	0x1	PD
18	0x0	TERM_EN
17:16	0x0	LS_FSLEW
15:14	0x0	LS_RSLEW
13:12	0x0	FS_SLEW
11:6	0x22	HS_SLEW
5:0	0x20	HS_CURR_LEVEL

19.13.12.43 XUSB_PADCTL_USB2_OTG_PAD2_CTL_0_0

OTGPAD2, CTL0 static settings

Offset: 0xa8 | Read/Write: R/W | Reset: 0x003808a0 (0bxxxxxxxx00111000000100010100000)

Bit	Reset	Description
23	0x0	LSBIAS_SEL
22	0x0	DISCON_DETECT_METHOD
21	0x1	PD_ZI
20	0x1	PD2
19	0x1	PD
18	0x0	TERM_EN
17:16	0x0	LS_FSLEW
15:14	0x0	LS_RSLEW
13:12	0x0	FS_SLEW
11:6	0x22	HS_SLEW
5:0	0x20	HS_CURR_LEVEL

19.13.12.44 XUSB_PADCTL_USB2_OTG_PAD0_CTL_1_0

OTGPAD0, CTL1 static settings

Offset: 0xac | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxx0000000000100)

Bit	Reset	Description
12:11	0x0	RPU_RANGE_ADJ
10:9	0x0	HS_IREF_CAP
8:7	0x0	SPARE
6:3	0x0	TERM_RANGE_ADJ
2	0x1	PD_DR
1	0x0	PD_DISC_FORCE_POWERUP
0	0x0	PD_CHRP_FORCE_POWERUP

19.13.12.45 XUSB_PADCTL_USB2_OTG_PAD1_CTL_1_0

OTGPAD1, CTL1 static settings

Offset: 0xb0 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxx0000000000100)

Bit	Reset	Description
12:11	0x0	RPU_RANGE_ADJ
10:9	0x0	HS_IREF_CAP
8:7	0x0	SPARE
6:3	0x0	TERM_RANGE_ADJ
2	0x1	PD_DR
1	0x0	PD_DISC_FORCE_POWERUP
0	0x0	PD_CHRP_FORCE_POWERUP

19.13.12.46 XUSB_PADCTL_USB2_OTG_PAD2_CTL_1_0

OTGPAD2, CTL1 static settings

Offset: 0xb4 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxx0000000000100)

Bit	Reset	Description
12:11	0x0	RPU_RANGE_ADJ
10:9	0x0	HS_IREF_CAP
8:7	0x0	SPARE
6:3	0x0	TERM_RANGE_ADJ
2	0x1	PD_DR
1	0x0	PD_DISC_FORCE_POWERUP
0	0x0	PD_CHRP_FORCE_POWERUP

19.13.12.47 XUSB_PADCTL_USB2_BIAS_PAD_CTL_0_0

Offset: 0xb8 | Read/Write: R/W | Reset: 0x00003000 (0bxxxxxxxxxxxxxxxx0001100000000000)

Bit	Reset	Description
16:14	0x0	ADJRPU
13	0x1	PD_TRK
12	0x1	PD
11:9	0x0	TERM_OFFSET
8:7	0x0	VBUS_LEVEL
6:5	0x0	HS_CHIRP_LEVEL
4:2	0x0	HS_DISCON_LEVEL
1:0	0x0	HS_SQUELCH_LEVEL

19.13.12.48 XUSB_PADCTL_USB2_BIAS_PAD_CTL_1_0

BIASPAD, CTL1 static settings

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

Bit	Reset	Description
31:16	X	TCTRL
15:0	X	RCTRL

19.13.12.49 XUSB_PADCTL_HSIC_PAD0_CTL_0_0

HSIC PAD0, CTL0

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000088 (0bxxxxxxxxxxxxx0000000000010001000)

Bit	Reset	Description
19:16	0x0	HSIC_OPT
15:12	0x0	TX_SLEWN
11:8	0x0	TX_SLEWP
7:4	0x8	TX_RTUNEN
3:0	0x8	TX_RTUNEP

19.13.12.50 XUSB_PADCTL_HSIC_PAD1_CTL_0_0

HSIC PAD1, CTL0

Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000088 (0bxxxxxxxxxxxxx0000000000010001000)

Bit	Reset	Description
19:16	0x0	HSIC_OPT
15:12	0x0	TX_SLEWN
11:8	0x0	TX_SLEWP
7:4	0x8	TX_RTUNEN
3:0	0x8	TX_RTUNEP

19.13.12.51 XUSB_PADCTL_HSIC_PAD0_CTL_1_0

HSIC PAD0, CTL1

Offset: 0xc8 | Read/Write: R/W | Reset: 0x000001b8 (0bxxxxxxxxxxxxxxxxxxxxxx00110111000)

Bit	Reset	Description
10	0x0	RPU_STROBE
9	0x0	RPU_DATA
8	0x1	RPD_STROBE
7	0x1	RPD_DATA
6	0x0	LPBK
5	0x1	PD_ZI
4	0x1	PD_RX
3	0x1	PD_TRX
2	0x0	PD_TX
1	0x0	IDDQ
0	0x0	AUTO_TERM_EN

19.13.12.52 XUSB_PADCTL_HSIC_PAD1_CTL_1_0

HSIC PAD1, CTL1

Offset: 0xcc | Read/Write: R/W | Reset: 0x000001b8 (0bxxxxxxxxxxxxxxxxxxxxxx00110111000)

Bit	Reset	Description
10	0x0	RPU_STROBE
9	0x0	RPU_DATA
8	0x1	RPD_STROBE
7	0x1	RPD_DATA
6	0x0	LPBK
5	0x1	PD_ZI
4	0x1	PD_RX
3	0x1	PD_TRX
2	0x0	PD_TX
1	0x0	IDDQ
0	0x0	AUTO_TERM_EN

19.13.12.53 XUSB_PADCTL_HSIC_PAD0_CTL_2_0

HSIC PAD0, CTL2

Offset: 0xd0 | Read/Write: R/W | Reset: 0xXXXX0022 (0bxxxxxxxxxxxxxxxxxxxxxxxx00100010)

Bit	R/W	Reset	Description
31:16	RO	X	CALIOUT

Bit	R/W	Reset	Description
7:4	RW	0x2	RX_STROBE_TRIM
3:0	RW	0x2	RX_DATA_TRIM

19.13.12.54 XUSB_PADCTL_HSIC_PAD1_CTL_2_0

HSIC PAD1, CTL2

Offset: 0xd4 | Read/Write: R/W | Reset: 0xXXXX0022 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00100010)

Bit	R/W	Reset	Description
31:16	RO	X	CALIOUT
7:4	RW	0x2	RX_STROBE_TRIM
3:0	RW	0x2	RX_DATA_TRIM

19.13.12.55 XUSB_PADCTL_ULPI_LINK_TRIM_CONTROL_0

ULPI LINK and NULL Mode Trimmer Controls

Offset: 0xd8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx0000000000xxxxx00x00000000)

Bit	Reset	Description
25	0x0	CTL_SEL_DEL1
24	0x0	CTL_SEL_DEL0
23:16	0x0	CTL_TRIM_VAL
10	0x0	DAT_SEL_DEL1
9	0x0	DAT_SEL_DEL0
7:0	0x0	DAT_TRIM_VAL

19.13.12.56 XUSB_PADCTL_ULPI_NULL_CLK_TRIM_CONTROL_0

Offset: 0xdc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx00000xxx00000)

Bit	Reset	Description
12:8	0x0	NULL_LBKCLK_TRIM_VAL
4:0	0x0	NULL_CLKOUT_TRIM_VAL

19.13.12.57 XUSB_PADCTL_HSIC_STRB_TRIM_CONTROL_0

HSIC STROBE Trimmer Control

Offset: 0xe0 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx000001)

Bit	Reset	Description
5:0	0x1	STRB_TRIM_VAL

19.13.12.58 XUSB_PADCTL_WAKE_CTRL_0

Wake Logic AUX RDET CLK FORCE ON

Offset: 0xe4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5	0x0	LANE_S0_FORCE_TX_RDET_CLK_ENABLE
4	0x0	LANE_P4_FORCE_TX_RDET_CLK_ENABLE
3	0x0	LANE_P3_FORCE_TX_RDET_CLK_ENABLE
2	0x0	LANE_P2_FORCE_TX_RDET_CLK_ENABLE
1	0x0	LANE_P 1_FORCE_TX_RDET_CLK_ENABLE
0	0x0	LANE_P 0_FORCE_TX_RDET_CLK_ENABLE

19.13.12.59 XUSB_PADCTL_PM_SPARE_0

PAD Macro Spare Bits

Offset: 0xe8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11	0x0	HSIC_PM_SPARE_BIT3
10	0x0	HSIC_PM_SPARE_BIT2
9	0x0	HSIC_PM_SPARE_BIT1
8	0x0	HSIC_PM_SPARE_BIT0
7	0x0	ULPI_PM_SPARE_BIT3
6	0x0	ULPI_PM_SPARE_BIT2
5	0x0	ULPI_PM_SPARE_BIT1
4	0x0	ULPI_PM_SPARE_BIT0
3	0x0	OTG_PM_SPARE_BIT3
2	0x0	OTG_PM_SPARE_BIT2
1	0x0	OTG_PM_SPARE_BIT1
0	0x0	OTG_PM_SPARE_BIT0

19.13.12.60 XUSB_PADCTL_IOPHY_MISC_PAD_P2_CTL_1_0

USB3 Pads : Miscellaneous Direction Control : The following lanes are only used by PCIe

USB3 Pads : Miscellaneous Ovrd : The following lanes are only used by PCIe

Offset: 0xec | Read/Write: R/W | Reset: 0x000aX330 (0bxxxx0000000010100xxx001100110000)

Bit	R/W	Reset	Description
27	RW	0x0	RX_PWR_OVRD
26	RW	0x0	TX_PWR_OVRD
25	RW	0x0	RATE_MODE_OVRD
24	RW	0x0	RATE_MODE

Bit	R/W	Reset	Description
23:22	RW	0x0	RX_DIV
21:20	RW	0x0	TX_DIV
19:18	RW	0x2	RX_RATE
17:16	RW	0x2	TX_RATE
15	RW	0x0	TX_RDET
13	RO	X	TX_STAT_PRESENT
12	RO	X	RX_STAT_IDLE
11	RW	0x0	RX_DATA_EN
10	RW	0x0	RX_DATA_READY
9:8	RW	0x3	RX_SLEEP
7	RW	0x0	TX_DATA_EN
6	RW	0x0	TX_DATA_READY
5:4	RW	0x3	TX_SLEEP
3	RW	0x0	CKBUFPD_OVRD
2	RW	0x0	CKBUFPD
1	RW	0x0	IDDQ_OVRD
0	RW	0x0	IDDQ

19.13.12.61 XUSB_PADCTL_IOPHY_MISC_PAD_P3_CTL_1_0

Offset: 0xf0 | Read/Write: R/W | Reset: 0x000aX330 (0bxxxx0000000010100xxx001100110000)

Bit	R/W	Reset	Description
27	RW	0x0	RX_PWR_OVRD
26	RW	0x0	TX_PWR_OVRD
25	RW	0x0	RATE_MODE_OVRD
24	RW	0x0	RATE_MODE
23:22	RW	0x0	RX_DIV
21:20	RW	0x0	TX_DIV
19:18	RW	0x2	RX_RATE
17:16	RW	0x2	TX_RATE
15	RW	0x0	TX_RDET
13	RO	X	TX_STAT_PRESENT
12	RO	X	RX_STAT_IDLE
11	RW	0x0	RX_DATA_EN
10	RW	0x0	RX_DATA_READY

Bit	R/W	Reset	Description
9:8	RW	0x3	RX_SLEEP
7	RW	0x0	TX_DATA_EN
6	RW	0x0	TX_DATA_READY
5:4	RW	0x3	TX_SLEEP
3	RW	0x0	CKBUFPD_OVRD
2	RW	0x0	CKBUFPD
1	RW	0x0	IDDQ_OVRD
0	RW	0x0	IDDQ

19.13.12.62 XUSB_PADCTL_IOPHY_MISC_PAD_P4_CTL_1_0

Offset: 0xf4 | Read/Write: R/W | Reset: 0x000aX330 (0bxxxx0000000010100xxx001100110000)

Bit	R/W	Reset	Description
27	RW	0x0	RX_PWR_OVRD
26	RW	0x0	TX_PWR_OVRD
25	RW	0x0	RATE_MODE_OVRD
24	RW	0x0	RATE_MODE
23:22	RW	0x0	RX_DIV
21:20	RW	0x0	TX_DIV
19:18	RW	0x2	RX_RATE
17:16	RW	0x2	TX_RATE
15	RW	0x0	TX_RDET
13	RO	X	TX_STAT_PRESENT
12	RO	X	RX_STAT_IDLE
11	RW	0x0	RX_DATA_EN
10	RW	0x0	RX_DATA_READY
9:8	RW	0x3	RX_SLEEP
7	RW	0x0	TX_DATA_EN
6	RW	0x0	TX_DATA_READY
5:4	RW	0x3	TX_SLEEP
3	RW	0x0	CKBUFPD_OVRD
2	RW	0x0	CKBUFPD
1	RW	0x0	IDDQ_OVRD
0	RW	0x0	IDDQ

19.13.12.63 XUSB_PADCTL_IOPHY_MISC_PAD_P2_CTL_2_0

Offset: 0xf8 | Read/Write: R/W | Reset: 0xX0000000 (0bxx000x00xxxxxxxxx00000000000000)

Bit	R/W	Reset	Description
31:30	RO	X	SPARE_OUT
29:28	RW	0x0	SPARE_IN
27	RW	0x0	TEST_EN
25	RW	0x0	PRBS_CHK_EN
24	RW	0x0	PRBS_ERROR
13	RW	0x0	RX_CDR_RESET
12	RW	0x0	TX_SYNC
11	RW	0x0	FED_LOOP
10:8	RW	0x0	TX_DATA_MODE
7	RW	0x0	FEA_LOOP
6:4	RW	0x0	FEA_MODE
3	RW	0x0	NEA_LOOP
2	RW	0x0	NED_LOOP
1:0	RW	0x0	NED_MODE

19.13.12.64 XUSB_PADCTL_IOPHY_MISC_PAD_P3_CTL_2_0

Offset: 0xfc | Read/Write: R/W | Reset: 0xX0000000 (0bxx000x00xxxxxxxxx00000000000000)

Bit	R/W	Reset	Description
31:30	RO	X	SPARE_OUT
29:28	RW	0x0	SPARE_IN
27	RW	0x0	TEST_EN
25	RW	0x0	PRBS_CHK_EN
24	RW	0x0	PRBS_ERROR
13	RW	0x0	RX_CDR_RESET
12	RW	0x0	TX_SYNC
11	RW	0x0	FED_LOOP
10:8	RW	0x0	TX_DATA_MODE
7	RW	0x0	FEA_LOOP
6:4	RW	0x0	FEA_MODE
3	RW	0x0	NEA_LOOP
2	RW	0x0	NED_LOOP
1:0	RW	0x0	NED_MODE

19.13.12.65 XUSB_PADCTL_IOPHY_MISC_PAD_P4_CTL_2_0

Offset: 0x100 | Read/Write: R/W | Reset: 0xX0000000 (0bxx000x00xxxxxxxxx00000000000000)

Bit	R/W	Reset	Description
31:30	RO	X	SPARE_OUT
29:28	RW	0x0	SPARE_IN
27	RW	0x0	TEST_EN
25	RW	0x0	PRBS_CHK_EN
24	RW	0x0	PRBS_ERROR
13	RW	0x0	RX_CDR_RESET
12	RW	0x0	TX_SYNC
11	RW	0x0	FED_LOOP
10:8	RW	0x0	TX_DATA_MODE
7	RW	0x0	FEA_LOOP
6:4	RW	0x0	FEA_MODE
3	RW	0x0	NEA_LOOP
2	RW	0x0	NED_LOOP
1:0	RW	0x0	NED_MODE

19.13.12.66 XUSB_PADCTL_IOPHY_MISC_PAD_P2_CTL_3_0

Offset: 0x104 | Read/Write: R/W | Reset: 0x00040200 (0b00000000000001000000010xxxx0000)

Bit	Reset	Description
31:20	0x0	CDR_TEST
19	0x0	RX_IDLE_MODE_OVRD
18	0x1	RX_IDLE_MODE
17	0x0	RX_IDLE_BYP
16	0x0	TX_RDET_BYP
15:14	0x0	RX_IDLE_T
13:12	0x0	TX_RDET_T
11:8	0x2	TX_SEL_LOAD
3:0	0x0	MISC_CNTL

19.13.12.67 XUSB_PADCTL_IOPHY_MISC_PAD_P3_CTL_3_0

Offset: 0x108 | Read/Write: R/W | Reset: 0x00040200 (0b00000000000001000000010xxxx0000)

Bit	Reset	Description
31:20	0x0	CDR_TEST

Bit	Reset	Description
19	0x0	RX_IDLE_MODE_OVRD
18	0x1	RX_IDLE_MODE
17	0x0	RX_IDLE_BYP
16	0x0	TX_RDET_BYP
15:14	0x0	RX_IDLE_T
13:12	0x0	TX_RDET_T
11:8	0x2	TX_SEL_LOAD
3:0	0x0	MISC_CNTL

19.13.12.68 XUSB_PADCTL_IOPHY_MISC_PAD_P4_CTL_3_0

Offset: 0x10c | Read/Write: R/W | Reset: 0x00040200 (0b00000000000001000000010xxxx0000)

Bit	Reset	Description
31:20	0x0	CDR_TEST
19	0x0	RX_IDLE_MODE_OVRD
18	0x1	RX_IDLE_MODE
17	0x0	RX_IDLE_BYP
16	0x0	TX_RDET_BYP
15:14	0x0	RX_IDLE_T
13:12	0x0	TX_RDET_T
11:8	0x2	TX_SEL_LOAD
3:0	0x0	MISC_CNTL

19.13.12.69 XUSB_PADCTL_IOPHY_MISC_PAD_P2_CTL_4_0

Offset: 0x110 | Read/Write: R/W | Reset: 0xXX000XX0 (0bx100x0000000xxxxxx0000x000x0xxxx)

Bit	R/W	Reset	Description
31	RO	X	AUX_RX_STAT_IDLE
30	RW	0x1	AUX_RX_IDLE_MODE
29	RW	0x0	AUX_RX_IDLE_EN
28	RW	0x0	AUX_RX_TERM_EN
27	RO	X	AUX_TX_STAT_PRESENT
26	RW	0x0	AUX_TX_RDET_CLK_EN
25	RW	0x0	AUX_TX_RDET_EN
24	RW	0x0	AUX_TX_TERM_EN
23	RW	0x0	AUX_MODE_OVRD
22	RW	0x0	AUX_HOLD_EN

Bit	R/W	Reset	Description
21	RW	0x0	AUX_IDDQ_OVRD
20	RW	0x0	AUX_IDDQ
13	RW	0x0	TX_BYP_OVRD
12	RW	0x0	RX_BYP_MODE
11	RW	0x0	RX_BYP_EN
10	RW	0x0	RX_BYP_DIR
9	RO	X	RX_BYP_IN
8	RW	0x0	RX_BYP_OUT
7	RW	0x0	TX_BYP_EN
6	RW	0x0	TX_BYP_DIR
5	RO	X	TX_BYP_IN
4	RW	0x0	TX_BYP_OUT

19.13.12.70 XUSB_PADCTL_IOPHY_MISC_PAD_P3_CTL_4_0

Offset: 0x114 | Read/Write: R/W | Reset: 0xXX000XX0 (0bx100x0000000xxxxxx00000x00x0xxxx)

Bit	R/W	Reset	Description
31	RO	X	AUX_RX_STAT_IDLE
30	RW	0x1	AUX_RX_IDLE_MODE
29	RW	0x0	AUX_RX_IDLE_EN
28	RW	0x0	AUX_RX_TERM_EN
27	RO	X	AUX_TX_STAT_PRESENT
26	RW	0x0	AUX_TX_RDET_CLK_EN
25	RW	0x0	AUX_TX_RDET_EN
24	RW	0x0	AUX_TX_TERM_EN
23	RW	0x0	AUX_MODE_OVRD
22	RW	0x0	AUX_HOLD_EN
21	RW	0x0	AUX_IDDQ_OVRD
20	RW	0x0	AUX_IDDQ
13	RW	0x0	TX_BYP_OVRD
12	RW	0x0	RX_BYP_MODE
11	RW	0x0	RX_BYP_EN
10	RW	0x0	RX_BYP_DIR
9	RW	0x0	RX_BYP_OUT
8	RO	X	RX_BYP_IN

Bit	R/W	Reset	Description
7	RW	0x0	TX_BYP_EN
6	RW	0x0	TX_BYP_DIR
5	RO	X	TX_BYP_IN
4	RW	0x0	TX_BYP_OUT

19.13.12.71 XUSB_PADCTL_IOPHY_MISC_PAD_P4_CTL_4_0

Offset: 0x118 | Read/Write: R/W | Reset: 0xXX000XX0 (0bx100x0000000xxxxxx00000x00x0xxxx)

Bit	R/W	Reset	Description
31	RO	X	AUX_RX_STAT_IDLE
30	RW	0x1	AUX_RX_IDLE_MODE
29	RW	0x0	AUX_RX_IDLE_EN
28	RW	0x0	AUX_RX_TERM_EN
27	RO	X	AUX_TX_STAT_PRESENT
26	RW	0x0	AUX_TX_RDET_CLK_EN
25	RW	0x0	AUX_TX_RDET_EN
24	RW	0x0	AUX_TX_TERM_EN
23	RW	0x0	AUX_MODE_OVRD
22	RW	0x0	AUX_HOLD_EN
21	RW	0x0	AUX_IDDQ_OVRD
20	RW	0x0	AUX_IDDQ
13	RW	0x0	TX_BYP_OVRD
12	RW	0x0	RX_BYP_MODE
11	RW	0x0	RX_BYP_EN
10	RW	0x0	RX_BYP_DIR
9	RW	0x0	RX_BYP_OUT
8	RO	X	RX_BYP_IN
7	RW	0x0	TX_BYP_EN
6	RW	0x0	TX_BYP_DIR
5	RO	X	TX_BYP_IN
4	RW	0x0	TX_BYP_OUT

19.13.12.72 XUSB_PADCTL_IOPHY_MISC_PAD_P2_CTL_5_0

Offset: 0x11c | Read/Write: R/W | Reset: 0x000XX0XX (0bxxxxxxxxxxxxxxxxxxxxxxxx00xx00xx0)

Bit	R/W	Reset	Description
17:12	RO	X	RX_QEYE_OUT

Bit	R/W	Reset	Description
8	RW	0x0	RX_QEYE_EN
7	RW	0x0	EOM_EN
5	RO	X	EOM_TRAIN_DONE
4	RW	0x0	EOM_TRAIN_EN
3	RW	0x0	DFE_RESET
1	RO	X	DFE_TRAIN_DONE
0	RW	0x0	DFE_TRAIN_EN

19.13.12.73 XUSB_PADCTL_IOPHY_MISC_PAD_P3_CTL_5_0

Offset: 0x120 | Read/Write: R/W | Reset: 0x000XX0XX (0bxxxxxxxxxxxxxxxxxxxxxxxx00xx00xx0)

Bit	R/W	Reset	Description
17:12	RO	X	RX_QEYE_OUT
8	RW	0x0	RX_QEYE_EN
7	RW	0x0	EOM_EN
5	RO	X	EOM_TRAIN_DONE
4	RW	0x0	EOM_TRAIN_EN
3	RW	0x0	DFE_RESET
1	RO	X	DFE_TRAIN_DONE
0	RW	0x0	DFE_TRAIN_EN

19.13.12.74 XUSB_PADCTL_IOPHY_MISC_PAD_P4_CTL_5_0

Offset: 0x124 | Read/Write: R/W | Reset: 0x000XX0XX (0bxxxxxxxxxxxxxxxxxxxxxxxx00xx00xx0)

Bit	R/W	Reset	Description
17:12	RO	X	RX_QEYE_OUT
8	RW	0x0	RX_QEYE_EN
7	RW	0x0	EOM_EN
5	RO	X	EOM_TRAIN_DONE
4	RW	0x0	EOM_TRAIN_EN
3	RW	0x0	DFE_RESET
1	RO	X	DFE_TRAIN_DONE
0	RW	0x0	DFE_TRAIN_EN

19.13.12.75 XUSB_PADCTL_IOPHY_MISC_PAD_P2_CTL_6_0

Offset: 0x128 | Read/Write: R/W | Reset: 0xXX000000 (0bxxxxxxxx00000000000000000000000)

Bit	R/W	Reset	Description
31:24	RO	X	MISC_OUT
23:16	RW	0x0	MISC_OUT_SEL
15:0	RW	0x0	MISC_TEST

19.13.12.76 XUSB_PADCTL_IOPHY_MISC_PAD_P3_CTL_6_0

Offset: 0x12c | Read/Write: R/W | Reset: 0xXX000000 (0bxxxxxxxx00000000000000000000000)

Bit	R/W	Reset	Description
31:24	RO	X	MISC_OUT
23:16	RW	0x0	MISC_OUT_SEL
15:0	RW	0x0	MISC_TEST

19.13.12.77 XUSB_PADCTL_IOPHY_MISC_PAD_P4_CTL_6_0

Offset: 0x130 | Read/Write: R/W | Reset: 0xXX000000 (0bxxxxxxxx00000000000000000000000)

Bit	R/W	Reset	Description
31:24	RO	X	MISC_OUT
23:16	RW	0x0	MISC_OUT_SEL
15:0	RW	0x0	MISC_TEST

19.13.12.78 XUSB_PADCTL_USB3_PAD_MUX_0

Offset: 0x134 | Read/Write: R/W | Reset: 0x08000000 (0bxxxx100000000000xxxxxxxx0000000)

Bit	Reset	Description
27:26	0x2	SATA_PAD_LANE0: 0 = PCIE 1 = USB3_SS 2 = SATA 3 = RESERVED
25:24	0x0	PCIE_PAD_LANE4: 0 = PCIE 1 = USB3_SS 2 = SATA 3 = RESERVED
23:22	0x0	PCIE_PAD_LANE3: 0 = PCIE 1 = USB3_SS 2 = SATA 3 = RESERVED
21:20	0x0	PCIE_PAD_LANE2: 0 = PCIE 1 = USB3_SS 2 = SATA 3 = RESERVED
19:18	0x0	PCIE_PAD_LANE1: 0 = PCIE 1 = USB3_SS 2 = SATA

Bit	Reset	Description
		3 = RESERVED
17:16	0x0	PCIE_PAD_LANE0: 0 = PCIE 1 = USB3_SS 2 = SATA 3 = RESERVED
6	0x0	FORCE_SATA_PAD_IDDQ_DISABLE_MASK0: 0 = NOT_DISABLED 1 = DISABLED
5	0x0	FORCE_PCIE_PAD_IDDQ_DISABLE_MASK4: 0 = NOT_DISABLED 1 = DISABLED
4	0x0	FORCE_PCIE_PAD_IDDQ_DISABLE_MASK3: 0 = NOT_DISABLED 1 = DISABLED
3	0x0	FORCE_PCIE_PAD_IDDQ_DISABLE_MASK2: 0 = NOT_DISABLED 1 = DISABLED
2	0x0	FORCE_PCIE_PAD_IDDQ_DISABLE_MASK1: 0 = NOT_DISABLED 1 = DISABLED
1	0x0	FORCE_PCIE_PAD_IDDQ_DISABLE_MASK0: 0 = NOT_DISABLED 1 = DISABLED
0	0x0	FORCE_PCIE_PAD_IDDQ_DISABLE: 0 = NOT_DISABLED 1 = DISABLED

19.13.12.79 XUSB_PADCTL_IOPHY_PLL_S0_CTL1_0

USB3/SATA PLL0 control signals : One set per PLL

Offset: 0x138 | Read/Write: R/W | Reset: 0x3X3X0000 (0bxx11xxx0xx11xxx000000x0000000000)

Bit	R/W	Reset	Description
29:28	RW	0x3	PLL1_REFCLK_NDIV
27	RO	X	PLL1_LOCKDET
24	RW	0x0	PLL1_MODE
21:20	RW	0x3	PLL0_REFCLK_NDIV
19	RO	X	PLL0_LOCKDET
16	RW	0x0	PLL0_MODE
15:12	RW	0x0	REFCLK_SEL
11	RW	0x0	REFCLK_TERM100
9	RW	0x0	PLL_CKBUFPD_OVR
8	RW	0x0	PLL_CKBUFPD_M
7	RW	0x0	PLL_CKBUFPD_BL

Bit	R/W	Reset	Description
6	RW	0x0	PLL_CKBUFPD_BR
5	RW	0x0	PLL_CKBUFPD_TL
4	RW	0x0	PLL_CKBUFPD_TR
3	RW	0x0	PLL_PWR_OVRD
2	RW	0x0	PLL_EMULATION_RST_
1	RW	0x0	PLL_RST_
0	RW	0x0	PLL_IDDQ

19.13.12.80 XUSB_PADCTL_IOPHY_PLL_S0_CTL2_0

Offset: 0x13c | Read/Write: R/W | Reset: 0xXX441020 (0bxxxxxxxx010001000x01000000100000)

Bit	R/W	Reset	Description
31:24	RO	X	PLL_MISC_OUT
23:20	RW	0x4	PLL1_CP_CNTL
19:16	RW	0x4	PLL0_CP_CNTL
15	RW	0x0	PLL_BYPASS_EN
13	RW	0x0	PLL_EMULATION_ON
12	RW	0x1	TCLKOUT_EN
11:8	RW	0x0	TCLKOUT_SEL
7	RW	0x0	XDIGCLK4P5_EN
6	RW	0x0	REFCLKBUF_EN
5	RW	0x1	TXCLKREF_EN
4	RW	0x0	TXCLKREF_SEL
3	RW	0x0	XDIGCLK_EN
2:0	RW	0x0	XDIGCLK_SEL

19.13.12.81 XUSB_PADCTL_IOPHY_PLL_S0_CTL3_0

Offset: 0x140 | Read/Write: R/W | Reset: 0x0000XX80 (0b0000xx000000xx00x0xxxxxx1xx00000)

Bit	R/W	Reset	Description
31:28	RW	0x0	PLL_TEMP_CNTL
25:20	RW	0x0	PLL_BW_CNTL
17:16	RW	0x0	PLL_BGAP_CNTL
15	RO	X	RCAL_DONE
14	RW	0x0	RCAL_RESET
12:8	RO	X	RCAL_VAL
7	RW	0x1	RCAL_BYPASS

Bit	R/W	Reset	Description
4:0	RW	0x0	RCAL_CODE

19.13.12.82 XUSB_PADCTL_IOPHY_PLL_S0_CTL4_0

Offset: 0x144 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PLL_MISC_CNTL

19.13.12.83 XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_1_0

Miscellaneous controls for USB3/SATA pads

Offset: 0x148 | Read/Write: R/W | Reset: 0x000aX330 (0bxxxx0000000010100xxx001100110000)

Bit	R/W	Reset	Description
27	RW	0x0	RX_PWR_OVRD
26	RW	0x0	TX_PWR_OVRD
25	RW	0x0	RATE_MODE_OVRD
24	RW	0x0	RATE_MODE
23:22	RW	0x0	RX_DIV
21:20	RW	0x0	TX_DIV
19:18	RW	0x2	RX_RATE
17:16	RW	0x2	TX_RATE
15	RW	0x0	TX_RDET
13	RO	X	TX_STAT_PRESENT
12	RO	X	RX_STAT_IDLE
11	RW	0x0	RX_DATA_EN
10	RW	0x0	RX_DATA_READY
9:8	RW	0x3	RX_SLEEP
7	RW	0x0	TX_DATA_EN
6	RW	0x0	TX_DATA_READY
5:4	RW	0x3	TX_SLEEP
3	RW	0x0	CKBUFPD_OVRD
2	RW	0x0	CKBUFPD
1	RW	0x0	IDDQ_OVRD
0	RW	0x0	IDDQ

19.13.12.84 XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_2_0

Offset: 0x14c | Read/Write: R/W | Reset: 0xX0000000 (0bxx000x00xxxxxxxxxx00000000000000)

Bit	R/W	Reset	Description
31:30	RO	X	SPARE_OUT
29:28	RW	0x0	SPARE_IN
27	RW	0x0	TEST_EN
25	RW	0x0	PRBS_CHK_EN
24	RW	0x0	PRBS_ERROR
13	RW	0x0	RX_CDR_RESET
12	RW	0x0	TX_SYNC
11	RW	0x0	FED_LOOP
10:8	RW	0x0	TX_DATA_MODE
7	RW	0x0	FEA_LOOP
6:4	RW	0x0	FEA_MODE
3	RW	0x0	NEA_LOOP
2	RW	0x0	NED_LOOP
1:0	RW	0x0	NED_MODE

19.13.12.85 XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_3_0

Offset: 0x150 | Read/Write: R/W | Reset: 0x00040200 (0b00000000000000010000000010xxxx0000)

Bit	Reset	Description
31:20	0x0	CDR_TEST
19	0x0	RX_IDLE_MODE_OVRD
18	0x1	RX_IDLE_MODE
17	0x0	RX_IDLE_BYP
16	0x0	TX_RDET_BYP
15:14	0x0	RX_IDLE_T
13:12	0x0	TX_RDET_T
11:8	0x2	TX_SEL_LOAD
3:0	0x0	MISC_CNTL

19.13.12.86 XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_4_0

Offset: 0x154 | Read/Write: R/W | Reset: 0xXX000XX0 (0bx100x0000000xxxxxx0000x000x0xxxx)

Bit	R/W	Reset	Description
31	RO	X	AUX_RX_STAT_IDLE
30	RW	0x1	AUX_RX_IDLE_MODE
29	RW	0x0	AUX_RX_IDLE_EN
28	RW	0x0	AUX_RX_TERM_EN

Bit	R/W	Reset	Description
27	RO	X	AUX_TX_STAT_PRESENT
26	RW	0x0	AUX_TX_RDET_CLK_EN
25	RW	0x0	AUX_TX_RDET_EN
24	RW	0x0	AUX_TX_TERM_EN
23	RW	0x0	AUX_MODE_OVRD
22	RW	0x0	AUX_HOLD_EN
21	RW	0x0	AUX_IDDQ_OVRD
20	RW	0x0	AUX_IDDQ
13	RW	0x0	TX_BYP_OVRD
12	RW	0x0	RX_BYP_MODE
11	RW	0x0	RX_BYP_EN
10	RW	0x0	RX_BYP_DIR
9	RO	X	RX_BYP_IN
8	RW	0x0	RX_BYP_OUT
7	RW	0x0	TX_BYP_EN
6	RW	0x0	TX_BYP_DIR
5	RO	X	TX_BYP_IN
4	RW	0x0	TX_BYP_OUT

19.13.12.87 XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_5_0

Offset: 0x158 | Read/Write: R/W | Reset: 0x000XX0XX (0bxxxxxxxxxxxxxxxxxxxxxxxx00xx00xx0)

Bit	R/W	Reset	Description
17:12	RO	X	RX_QEYE_OUT
8	RW	0x0	RX_QEYE_EN
7	RW	0x0	EOM_EN
5	RO	X	EOM_TRAIN_DONE
4	RW	0x0	EOM_TRAIN_EN
3	RW	0x0	DFE_RESET
1	RO	X	DFE_TRAIN_DONE
0	RW	0x0	DFE_TRAIN_EN

19.13.12.88 XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_6_0

Offset: 0x15c | Read/Write: R/W | Reset: 0xXX000000 (0bxxxxxxxx00000000000000000000000)

Bit	R/W	Reset	Description
31:24	RO	X	MISC_OUT

Bit	R/W	Reset	Description
23:16	RW	0x0	MISC_OUT_SEL
15:0	RW	0x0	MISC_TEST

19.13.13 XUSB PCI Config Registers

19.13.13.1 T_XUSB_CFG_0

Offset: 0x00 | Read/Write: R

Bits	Reset	R/W	Description
31:16	FA3h	R	T_XUSB_CFG_0_DEVICE_ID_UNIT: FA3h: DEVICE_ID_UNIT_XUSB (default)
15:0	10DEh	R	T_XUSB_CFG_0_VENDOR_ID: 10DEh: VENDOR_ID_NVIDIA (default)

19.13.13.2 T_XUSB_CFG_1

FPCI Configuration Register

Offset: 0x04 | Read/Write: R/W

Bits	Reset	R/W	Description
26:25	0	R	T_XUSB_CFG_1_DEVSEL_TIMING: 0h: DEVSEL_TIMING_FAST (default) 1h: DEVSEL_TIMING_MEDIUM 2h: DEVSEL_TIMING_SLOW
31	0	R	T_XUSB_CFG_1_DETECTED_PERR: 0h: DETECTED_PERR_NOT_ACTIVE (default) 1h: DETECTED_PERR_ACTIVE 1h: DETECTED_PERR_CLEAR
30	0	R	T_XUSB_CFG_1_SIGNALED_SERR: 0h: SIGNALED_SERR_NOT_ACTIVE (default) 1h: SIGNALED_SERR_ACTIVE 1h: SIGNALED_SERR_CLEAR
18:11	0	R	Reserved
29	0	R	T_XUSB_CFG_1_RECEIVED_MASTER: 0h: RECEIVED_MASTER_NO_ABORT (default) 1h: RECEIVED_MASTER_ABORT 1h: RECEIVED_MASTER_CLEAR
28	0	R	T_XUSB_CFG_1_RECEIVED_TARGET: 0h: RECEIVED_TARGET_NO_ABORT (default) 1h: RECEIVED_TARGET_ABORT 1h: RECEIVED_TARGET_CLEAR
27	0	R	T_XUSB_CFG_1_SIGNALED_TARGET: 0h: SIGNALED_TARGET_NO_ABORT (default) 1h: SIGNALED_TARGET_ABORT 1h: SIGNALED_TARGET_CLEAR
24	0	R	T_XUSB_CFG_1_MASTER_DATA_PERR: 0h: MASTER_DATA_PERR_NOT_ACTIVE (default) 1h: MASTER_DATA_PERR_ACTIVE 1h: MASTER_DATA_PERR_CLEAR
23	1h	R	T_XUSB_CFG_1_FAST_BACK2BACK: 0h: FAST_BACK2BACK_INCAPABLE 1h: FAST_BACK2BACK_CAPABLE (default)

Bits	Reset	R/W	Description
22	0	R	Reserved
21	1h	R	T_XUSB_CFG_1_66MHZ: 0h: 66MHZ_INCAPABLE 1h: 66MHZ_CAPABLE (default)
20	1h	R	T_XUSB_CFG_1_CAPLIST: 0h: CAPLIST_NOT_PRESENT 1h: CAPLIST_PRESENT (default)
19	None	R	T_XUSB_CFG_1_INTR_STATUS: 0h: INTR_STATUS_0 1h: INTR_STATUS_1
10	0	R/W	T_XUSB_CFG_1_INTR_DISABLE: 0h: INTR_DISABLE_ON (default) 1h: INTR_DISABLE_OFF
9	0	R	T_XUSB_CFG_1_BACK2BACK: 0h: BACK2BACK_DISABLED (default) 1h: BACK2BACK_ENABLED
8	0	R	T_XUSB_CFG_1_SERR: 0h: SERR_DISABLED (default) 1h: SERR_ENABLED
7	0	R	T_XUSB_CFG_1_STEP: 0h: STEP_DISABLED (default) 1h: STEP_ENABLED
6	0	R	T_XUSB_CFG_1_PERR: 0h: PERR_DISABLED (default) 1h: PERR_ENABLED
5	0	R	T_XUSB_CFG_1_PALETTE_SNOOP: 0h: PALETTE_SNOOP_DISABLED (default) 1h: PALETTE_SNOOP_ENABLED
4	0	R	T_XUSB_CFG_1_WRITE_AND_INVAL: 0h: WRITE_AND_INVAL_DISABLED (default) 1h: WRITE_AND_INVAL_ENABLED
3	0	R	T_XUSB_CFG_1_SPECIAL_CYCLE: 0h: SPECIAL_CYCLE_DISABLED (default) 1h: SPECIAL_CYCLE_ENABLED
2	0	R/W	T_XUSB_CFG_1_BUS_MASTER: 0h: BUS_MASTER_DISABLED (default) 1h: BUS_MASTER_ENABLED
1	0	R/W	T_XUSB_CFG_1_MEMORY_SPACE: 0h: MEMORY_SPACE_DISABLED (default) 1h: MEMORY_SPACE_ENABLED
0	0	R/W	T_XUSB_CFG_1_IO_SPACE: 0h: IO_SPACE_DISABLED (default) 1h: IO_SPACE_ENABLED

19.13.13.3 T_XUSB_CFG_2

PCI Revision ID and Class Code Register

Offset: 0x08 | Read/Write: R

Bits	Reset	R/W	Description
31:24	Ch	R	T_XUSB_CFG_2_BASE_CLASS: The CLASS_CODE bits identify the generic function of the device and (in some cases) a specific register-level programming interface. The register is broken into three byte-size fields. The upper byte (at offset 0BH) is a base class code which broadly classifies the type of function the device

Bits	Reset	R/W	Description
			performs. The middle-byte (at offset 0BH) is a sub-class code which identifies more specifically the function of the device. The lower byte (at offset 09H) identifies a specific register-level programming interface (if any) so that device independent software can interact with the device. The Class Code and Revision ID are defined by parameters per block. Ch: BASE_CLASS_SBC (default)
23:16	3h	R	T_XUSB_CFG_2_SUB_CLASS: 3h: SUB_CLASS_XUSB (default)
15:8	30h	R	T_XUSB_CFG_2_PROG_IF: 30h: PROG_IF_XHCI (default)
7:0	A1h	R	T_XUSB_CFG_2_REVISION_ID: The REVISION_ID bits specify a device specific revision identifier. The value is chosen by the vendor. Zero is an acceptable value. This field should be viewed as a vendor defined extension to the DEVICE_ID. A1h: REVISION_ID_VAL (default) A1h: REVISION_ID_A01

19.13.13.4 T_XUSB_CFG_3

PCI Configuration Register

Offset: 0x0C | Read/Write: R

Bits	Reset	R/W	Description
31:24	0	R	Reserved
23	0	R	T_XUSB_CFG_3_HEADER_TYPE_FUNC: 0h: HEADER_TYPE_FUNC_SINGLE (default) 1h: HEADER_TYPE_FUNC_MULTI
22:16	0	R	T_XUSB_CFG_3_HEADER_TYPE_DEVICE: The HEADER_TYPE bits identify the layout of the bytes 10h through 3Fh in configuration space and also whether or not the device contains multiple functions. Bit 7 in this register is used to identify a multi-function device. If the bit is 0, then the device is single function. If the bit is 1, then the device has multiple functions. Bits 6 through 0 specify the layout of bytes 10h through 3Fh. The LATENCY_TIMER and HEADER_TYPE are defined by parameters per block. 0h: HEADER_TYPE_DEVICE_NON_BRIDGE (default) 1h: HEADER_TYPE_DEVICE_P2P_BRIDGE
15:11	0	R	T_XUSB_CFG_3_LATENCY_TIMER: The LATENCY_TIMER bits contain, in units of PCI bus clocks, the value of the Latency Timer for this PCI bus master. This register must be implemented as writable by any master that can burst more than two data phases. This register may be implemented as read-only for devices that burst two or fewer data phases, but the hardwired value must be limited to 16 or less. A typical implementation would be to build the five high-order bits (leaving the bottom three as read-only), resulting in a timer granularity of eight clocks. At reset, the register should be set to 0 (if programmable). LATENCY_TIMER bits are writable. 0h: LATENCY_TIMER_0_CLOCKS (default) 1h: LATENCY_TIMER_8_CLOCKS 1Eh: LATENCY_TIMER_240_CLOCKS 1Fh: LATENCY_TIMER_248_CLOCKS
10:8	0	R	Reserved
7:0	0	R	T_XUSB_CFG_3_CACHE_LINE_SIZE: 0h: CACHE_LINE_SIZE_0 (default) 20h: CACHE_LINE_SIZE_32 40h: CACHE_LINE_SIZE_64

19.13.13.5 T_XUSB_CFG_4

PCI Configuration Register

Offset: 0x10 | Read/Write: R/W

Bits	Reset	R/W	Description
31:15	0	R/W	T_XUSB_CFG_4_BASE_ADDRESS: The BASE_ADDRESS bits contain the base address of the device. The number of upper bits that a device actually implements depends on how much of the address space the device will respond to. A device that wants a 1 MB memory address space (using a 32-bit base address register) would build the top 12 bits of the address register, hardwiring the other bits to 0. Power-up software can determine how much address space the device required by writing a value of all 1's to the register and then reading the value back. The device will return 0's in all don't-care address bits, effectively specifying the address space required. 0h: BASE_ADDRESS_DEFAULT (default)
14:4	0	R	T_XUSB_CFG_4_BAR_SIZE_32KB: 0h: BAR_SIZE_32KB_RSVD (default)
3	1h	R	T_XUSB_CFG_4_PREFETCHABLE: 0h: PREFETCHABLE_NOT 1h: PREFETCHABLE_MERGABLE (default)
2:1	2h	R	T_XUSB_CFG_4_ADDRESS_TYPE: The ADDRESS_TYPE bits contain the type of the Base Address. It can be 32 bits, 20 bits, or 64 bits wide. 0h: ADDRESS_TYPE_32_BIT 2h: ADDRESS_TYPE_64_BIT (default)
0	0	R	T_XUSB_CFG_4_SPACE_TYPE: The SPACE_TYPE bit indicates whether the register maps into Memory or I/O space. 0h: SPACE_TYPE_MEMORY (default) 1h: SPACE_TYPE_IO

19.13.13.6 T_XUSB_CFG_5

Offset: 0x14 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_CFG_5_BASE_ADDRESSHI: 0h: BASE_ADDRESSHI_DEFAULT (default)

19.13.13.7 T_XUSB_CFG_6

Offset: 0x18-0x28 | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_CFG_6_RSVD: 0h: RSVD_00 (default)

19.13.13.8 T_XUSB_CFG_11

PCI Configuration Register

The SUBSYSTEM_VENDOR_ID bits and SUBSYSTEM_ID bits are used to uniquely identify the add-in board or subsystem where the device resides. When the device is on the motherboard, there is no serial ROM and the registers both initialize to NONE. The motherboard BIOS must set the values of the Subsystem ID and Subsystem Vendor ID by writing the proper values to the SUBSYSTEM_VENDOR_ID and SUBSYSTEM_ID bits in the PCI_T_16 register (NOT PCI_T_11).

Offset: 0x2c | Read/Write: R

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_CFG_11_SUBSYSTEM_ID: 0h: SUBSYSTEM_ID_NONE (default)
15:0	0	R	T_XUSB_CFG_11_SUBSYSTEM_VENDOR_ID: 0h: SUBSYSTEM_VENDOR_ID_NONE (default)

19.13.13.9 T_XUSB_CFG_12

PCI Configuration Register

Offset: 0x30 | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_CFG_12_RESERVED: 0h: RESERVED_0 (default)

19.13.13.10 T_XUSB_CFG_13

PCI Capability Pointer Register

Offset: 0x34 | Read/Write: R

Bits	Reset	R/W	Description
31:8	0	R	Reserved
7:0	44h	R	T_XUSB_CFG_13_CAP_PTR: The CAP_PTR bits indicate the offset into configuration space where the capabilities list begins. This always points to 0x44 where at least the PCI-PM registers are expected to reside. 44h: CAP_PTR_PMCAP C0h: CAP_PTR_MSI 70h: CAP_PTR_MSIX DCh: CAP_PTR_MMAP 44h: CAP_PTR_DEFAULT (default)

19.13.13.11 T_XUSB_CFG_14

PCI Configuration Register

Offset: 0x38 | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_CFG_14_RESERVED: 0h: RESERVED_0 (default)

19.13.13.12 T_XUSB_CFG_15

PCI Configuration Register

Offset: 0x3c | Read/Write: R/W

Bits	Reset	R/W	Description
31:24	0	R	T_XUSB_CFG_15_MAX_LAT: The MAX_LAT bits contain the maximum time the device requires to gain access to the CPI bus. This read-only register is used to specify the device's desired settings for Latency Timer values. The value specifies a period of time in units of 1/4 microseconds. Values of 0 indicate that the device has no major requirements for the settings of Latency Timers. MAX_LAT is nonzero. The INTR_PIN, MIN_GNT, and MAX_LAT are configurable per block. For a 64-byte buffer with two 32-byte sections, the maximum tolerable latency for the XHCI once a large XUSB ISO transaction has begun is about 23 µs. The latency timer is hardwired to 20 µs. This value only applies in External PCI operation. 0h: MAX_LAT_NO_REQUIREMENTS (default) 14h: MAX_LAT_5US 50h: MAX_LAT_20US

Bits	Reset	R/W	Description
23:16	0	R	T_XUSB_CFG_15_MIN_GNT: The MIN_GNT bits contain the length of the burst period a device needs assuming a clock rate of 33 MHz. This read-only register is used to specify the device's desired settings for Latency Timer values. The value specifies a period of time in units of 1/4 microsecond. Values of 0 indicate that the device has no major requirements for the settings of Latency Timers. MIN_GNT is nonzero. 0h: MIN_GNT_NO_REQUIREMENTS (default) 1h: MIN_GNT_240NS
15:8	1h	R	T_XUSB_CFG_15_INTR_PIN: The INTR_PIN bits contain the interrupt pin the device (or device function) uses. A value of 1 corresponds to INTA#. A value of 2 corresponds to INTB#. A value of 3 corresponds to INTC#. A value of 4 corresponds to INTD#. Devices (or device functions) that do not use an interrupt pin must put a 0 in this register. 0h: INTR_PIN_NONE 1h: INTR_PIN_INTA (default) 2h: INTR_PIN_INTB 3h: INTR_PIN_INTC 4h: INTR_PIN_INTD
7:0	0	R/W	T_XUSB_CFG_15_INTR_LINE: The INTR_LINE bits contain the interrupt routing information. The register is read/write and must be implemented by any device (or device function) that uses an interrupt pin. POST software will write the routing information into this register as it initializes and configures the system. The value in this register tells which input of the system interrupt controller(s) the device's interrupt pin is connected to. Device drivers and operating systems can use this information to determine priority and vector information. INTR_LINE is initialized to 0xff (no connection) at reset. Some PCI BIOS' cannot handle aliased INTR_LINEs. Some PCI BIOS' cannot handle INTR_LINE initialized to 0xff. 0h: INTR_LINE_IRQ0 (default) 1h: INTR_LINE_IRQ1 Fh: INTR_LINE_IRQ15 FFh: INTR_LINE_UNKNOWN

19.13.13.13 T_XUSB_CFG_16

Backdoor register write for updating subsystem ID/vendor ID.

Offset: 0x40 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R/W	T_XUSB_CFG_16_SUBSYSTEM_ID: 0h: SUBSYSTEM_ID_NONE (default)
15:0	0	R/W	T_XUSB_CFG_16_SUBSYSTEM_VENDOR_ID: 0h: SUBSYSTEM_VENDOR_ID_NONE (default)

19.13.13.14 T_XUSB_CFG_17

Offset: 0x44 | Read/Write: R

Bits	Reset	R/W	Description
31	1h	R	T_XUSB_CFG_17_D3CPME_SUPPORT: 1h: D3CPME_SUPPORT_YES 0h: D3CPME_SUPPORT_NO
30	1h	R	T_XUSB_CFG_17_D3HPME_SUPPORT: 1h: D3HPME_SUPPORT_YES 0h: D3HPME_SUPPORT_NO
29	0	R	T_XUSB_CFG_17_D2PME_SUPPORT: 1h: D2PME_SUPPORT_YES 0h: D2PME_SUPPORT_NO
28	0	R	T_XUSB_CFG_17_D1PME_SUPPORT:

Bits	Reset	R/W	Description
			1h: D1PME_SUPPORT_YES 0h: D1PME_SUPPORT_NO
27	1h	R	T_XUSB_CFG_17_D0PME_SUPPORT: 1h: D0PME_SUPPORT_YES 0h: D0PME_SUPPORT_NO
26	0	R	T_XUSB_CFG_17_D2_SUPPORT: 0h: D2_SUPPORT_NO 1h: D2_SUPPORT_YES
25	0	R	T_XUSB_CFG_17_D1_SUPPORT: 0h: D1_SUPPORT_NO 1h: D1_SUPPORT_YES
24:22	0	R	T_XUSB_CFG_17_AUXCUR: 0h: AUXCUR_SELF 1h: AUXCUR_55MA 2h: AUXCUR_100MA 3h: AUXCUR_160MA 4h: AUXCUR_220MA 5h: AUXCUR_270MA 6h: AUXCUR_320MA 7h: AUXCUR_375MA
21	0	R	T_XUSB_CFG_17_DSI: 0h: DSI_NONE 1h: DSI_NEEDED
20	0	R	T_XUSB_CFG_17_RSVD: 0h: RSVD_0
19	0	R	T_XUSB_CFG_17_PMECLK: 0h: PMECLK_NOT_REQUIRED 1h: PMECLK_REQUIRED
18:16	3h	R	T_XUSB_CFG_17_VER: 3h: VER_1P2
15:8	C0h	R	T_XUSB_CFG_17_NEXT_PTR: C0h: NEXT_PTR_MSI 70h: NEXT_PTR_MSIX DCh: NEXT_PTR_MMAP 0h: NEXT_PTR_NULL C0h: NEXT_PTR_DEFAULT (default)
7:0	1h	R	T_XUSB_CFG_17_CAP: 1h: CAP_PCIPM

19.13.13.15 T_XUSB_CFG_18_PMCSR

Offset: 0x48 | Read/Write: R/W

Bits	Reset	R/W	Description
31:24	0	R	Reserved
23:16	0	R	T_XUSB_CFG_18_PMCSR_BSE_RSVD: 0h: BSE_RSVD_00
15	0	RW1C	T_XUSB_CFG_18_PMCSR_PMESTATUS: 0h: PMESTATUS_NOT_PENDING (default) 1h: PMESTATUS_PENDING 1h: PMESTATUS_CLEAR
14:13	0	R	T_XUSB_CFG_18_PMCSR_DSCALE: 0h: DSCALE_INIT
12:9	0	R	T_XUSB_CFG_18_PMCSR_DSEL: 0h: DSEL_INIT

Bits	Reset	R/W	Description
8	0	R/W	T_XUSB_CFG_18_PMCSR_PME: 1h: PME_ENABLE 0h: PME_DISABLE (default)
7:4	0	R	T_XUSB_CFG_18_PMCSR_RSVD1: 0h: RSVD1_00
3	1h	R	T_XUSB_CFG_18_PMCSR_NSR: 1h: NSR_NORESET 0h: NSR_RESET
2	0	R	T_XUSB_CFG_18_PMCSR_RSVD0: 0h: RSVD0_0
1:0	0	R/W	T_XUSB_CFG_18_PMCSR_PWRSTATE: 0h: PWRSTATE_D0 (default) 1h: PWRSTATE_D1 2h: PWRSTATE_D2 3h: PWRSTATE_D3H

19.13.13.16 T_XUSB_CFG_24

XUSB XHCI Configuration Control

Offset: 0x60 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	Reserved
15:14	0	R	T_XUSB_CFG_24_RSVDP: 0h: RSVDP_VALUE
13:8	20h	R/W	T_XUSB_CFG_24_FLADJ: Frame Length Adjustment Register. This register is in the auxiliary power well. This feature is used to adjust any offset from the clock source that generates the clock that drives the SOF counter. When a new value is written into these six bits, the length of the frame is adjusted. Its initial programmed value is system dependent based on the accuracy of hardware USB clock and is initialized by system BIOS. This register should only be modified when the HChalted bit in the USBSTS register is a one. Changing value of this register while the host controller is operating yields undefined results. It should not be reprogrammed by USB system software unless the default or BIOS programmed values are incorrect, or the system is restoring the register while returning from a suspended state. 20h: FLADJ_VALUE (default)
7:0	30h	R	T_XUSB_CFG_24_SBRN: Serial Bus Release Number Register. This register contains the release of the Universal Serial Bus Specification with which this Universal Serial Bus Host Controller module is compliant. 30h: SBRN_VALUE

19.13.13.17 T_XUSB_CFG_EMU_RSVD

MSIX is not part of Tegra K1 processors.

The PCIe and HT capabilities are not required. PCIe capability will be trapped by the emulation only IP - the emulation only IP put the PCIe capability at 80h ~ BBh, and uses BCh ~ BFh, so these regions are reserved for the emulation only IP. (The emulation only IP will mask the pointers to insert the PCIe capability to the config space of the unit.)

Offset: 0x80-0xBB | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	Unknown	R/W	T_XUSB_CFG_EMU_RSVD_FIELD

19.13.13.18 T_XUSB_MSI_CTRL

MSI Message Control and Capability Register B0h

MSI_CTRL is the main control register for MSI support.

Offset: 0xc0 | Read/Write: R/W

Bits	Reset	R/W	Description
31:25	0	R	Reserved
24	0	R	T_XUSB_MSI_CTRL_VECTOR_MASK_CAP: The VECTOR_MASK_CAP field indicates whether or not the controller supports MSI-per-vector masking. 0h: VECTOR_MASK_CAP_DIS 1h: VECTOR_MASK_CAP_EN 0h: VECTOR_MASK_CAP_DEFAULT (default)
23	1h	R	T_XUSB_MSI_CTRL_64_ADDR_CAP: The 64_ADDR_CAP field indicates whether or not the controller is capable of generating a 64-bit message address. A value of 1 means the controller is capable of generating a 64-bit message address. 0h: 64_ADDR_CAP_DIS 1h: 64_ADDR_CAP_EN 1h: 64_ADDR_CAP_DEFAULT (default)
22:20	0	R/W	T_XUSB_MSI_CTRL_MULT_MSG_ENABLE: System software writes to this field to indicate the number of allocated vectors (less than or equal to the number of vectors requested). The number of vectors is aligned as a power of two. When MSI is enabled, the controller will be allocated at least one vector. 0h: MULT_MSG_ENABLE_1 1h: MULT_MSG_ENABLE_2 2h: MULT_MSG_ENABLE_4 3h: MULT_MSG_ENABLE_8 4h: MULT_MSG_ENABLE_16 5h: MULT_MSG_ENABLE_32 0h: MULT_MSG_ENABLE_DEFAULT (default)
19:17	0	R	T_XUSB_MSI_CTRL_MULT_MSG_CAP: System software reads this field to determine the number of requested vectors. The number of requested vectors must be aligned to a power of two. Values of 6 and 7 in this field are reserved. 0h: MULT_MSG_CAP_1 1h: MULT_MSG_CAP_2 2h: MULT_MSG_CAP_4 3h: MULT_MSG_CAP_8 4h: MULT_MSG_CAP_16 5h: MULT_MSG_CAP_32 0h: MULT_MSG_CAP_DEFAULT (default)
16	0	R/W	T_XUSB_MSI_CTRL_MSI_ENABLE: The MSI_ENABLE field enables the MSI capability. If MSI_ENABLE is written to a 1, the controller is permitted to use MSI to request service and is prohibited from using the legacy interrupt. System configuration software sets this bit to enable MSI. A device driver is prohibited from writing this bit to mask the controller's service request. If this bit is written to a 0, the controller is prohibited from using MSI to request service. 0h: MSI_ENABLE_OFF 1h: MSI_ENABLE_ON 0h: MSI_ENABLE_DEFAULT (default)
15:8	E0h	R	T_XUSB_MSI_CTRL_NEXT_PTR: The NEXT_PTR field identifies the next item in the capabilities list. It is a read-only field. 70h: NEXT_PTR_MSIX DCh: NEXT_PTR_MMAP 44h: NEXT_PTR_PMCAP

Bits	Reset	R/W	Description
			0h: NEXT_PTR_NULL E0h: NEXT_PTR_MAILBOX E0h: NEXT_PTR_DEFAULT (default)
7:0	5h	R	T_XUSB_MSI_CTRL_CAP_ID: The CAP_ID field identifies this capability block as the MSI capability block. This is read-only as 0x5. 5h: CAP_ID_MSI (default)

19.13.13.19 T_XUSB_MSI_ADDR1

MSI Message Address Register 84h

MSI_ADDR1 specifies the lower 32 bits to which an MSI memory write transaction should occur.

Offset: 0xc4 | Read/Write: R/W

Bits	Reset	R/W	Description
31:2	0	R/W	T_XUSB_MSI_ADDR1_MSG_ADDR: System-specified message address. When MSI is enabled, this field specifies the Dword-aligned address for the MSI memory write transaction. 0h: MSG_ADDR_DEFAULT (default)
1:0	0	R	Reserved

19.13.13.20 T_XUSB_MSI_ADDR2

MSI Message Upper Address Register 88h

MSI_ADDR2 specifies the upper 32 bits to which an MSI memory write transaction should occur.

Offset: 0xc8 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_MSI_ADDR2_MSG_ADDR: System-specified message address. When MSI is enabled, this field specifies the upper 32 bits of the address for the MSI memory write transaction. The contents of this register only apply when T_XUSB_CFG_MSI_CTRL_64_ADDR_CAP bit is set. 0h: MSG_ADDR_DEFAULT (default)

19.13.13.21 T_XUSB_MSI_DATA

MSI Message Data Register 8Ch

The MSI_DATA register contains the system-specified message.

Offset: 0xcc | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	Reserved
15:0	0	R/W	T_XUSB_MSI_DATA_MSG_DATA: System-specified message When MSI is enabled, the message data is driven onto the lower 16 bits of the MSI memory write. The MULT_MSG_ENABLE field in configuration register 80h specifies the number of low-order message data bits that the XHCI is permitted to modify to generate its system software allocated vectors. 0h: MSG_DATA_DEFAULT (default)

19.13.14 XUSB PCIe PADCTL REFCLK Configuration Registers

19.13.14.1 NV_PROJ__PCIE2_PADS_REFCLK_CFG0

Offset: 0xc8 | Read/Write: R/W

Bits	Reset	Description
31:28	0h	NV_PROJ__PCIE2_PADS_REFCLK_CFG0_REFCLK1_DRVI 0h: NV_PROJ__PCIE2_PADS_REFCLK_CFG0_REFCLK1_DRVI_DEFAULT
27:24	0h	NV_PROJ__PCIE2_PADS_REFCLK_CFG0_REFCLK1_PREDI 0h: NV_PROJ__PCIE2_PADS_REFCLK_CFG0_REFCLK1_PREDI_DEFAULT
23	0h	NV_PROJ__PCIE2_PADS_REFCLK_CFG0_REFCLK1_E_TERM 0h: NV_PROJ__PCIE2_PADS_REFCLK_CFG0_REFCLK1_E_TERM_DEFAULT
22:18	0h	NV_PROJ__PCIE2_PADS_REFCLK_CFG0_REFCLK1_TERM 0h: NV_PROJ__PCIE2_PADS_REFCLK_CFG0_REFCLK1_TERM_DEFAULT
15:12	0h	NV_PROJ__PCIE2_PADS_REFCLK_CFG0_REFCLK0_DRVI 0h: NV_PROJ__PCIE2_PADS_REFCLK_CFG0_REFCLK0_DRVI_DEFAULT
11:8	0h	NV_PROJ__PCIE2_PADS_REFCLK_CFG0_REFCLK0_PREDI 0h: NV_PROJ__PCIE2_PADS_REFCLK_CFG0_REFCLK0_PREDI_DEFAULT
7	0h	NV_PROJ__PCIE2_PADS_REFCLK_CFG0_REFCLK0_E_TERM 0h: NV_PROJ__PCIE2_PADS_REFCLK_CFG0_REFCLK0_E_TERM_DEFAULT
6:2	0h	NV_PROJ__PCIE2_PADS_REFCLK_CFG0_REFCLK0_TERM 0h: NV_PROJ__PCIE2_PADS_REFCLK_CFG0_REFCLK0_TERM_DEFAULT

19.13.14.2 NV_PROJ__PCIE2_PADS_REFCLK_CFG1

Offset: 0xcc | Read/Write: R/W

Bits	Reset	Description
15:12	0h	NV_PROJ__PCIE2_PADS_REFCLK_CFG1_REFCLK2_DRVI 0h: NV_PROJ__PCIE2_PADS_REFCLK_CFG1_REFCLK2_DRVI_DEFAULT Control for reference current on IBIAS outputs of PEX CLK COMP pad. **REF[2:0] IBIAS current output** 000 50 µA 001 75 µa 010 100 µA 011 125 µA 100 150 µA 101 200 µA 110 250 µA 111 300 µA
11:8	0h	NV_PROJ__PCIE2_PADS_REFCLK_CFG1_REFCLK2_PREDI 0h: NV_PROJ__PCIE2_PADS_REFCLK_CFG1_REFCLK2_PREDI_DEFAULT
7	0h	NV_PROJ__PCIE2_PADS_REFCLK_CFG1_REFCLK2_E_TERM 0h: NV_PROJ__PCIE2_PADS_REFCLK_CFG1_REFCLK2_E_TERM_DEFAULT
6:2	0h	NV_PROJ__PCIE2_PADS_REFCLK_CFG1_REFCLK2_TERM 0h: NV_PROJ__PCIE2_PADS_REFCLK_CFG1_REFCLK2_TERM_DEFAULT

19.13.15 XUSB Mailbox Registers

19.13.15.1 T_XUSB_CFG_ARU_MAILBOX_CAP

Offset: 0xe0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0b00000000000000000000000000000000)

Bits	Reset	R/W	Description
31:24	0	R/W	T_XUSB_CFG_ARU_MAILBOX_CAP_RSVD: 0h: MAILBOX_CAP_RSVD_INIT (default)
23:16	14h	RW	T_XUSB_CFG_ARU_MAILBOX_CAP_LENGTH: 14h: MAILBOX_CAP_LENGTH_INIT (default)
15:8	0	RW	T_XUSB_CFG_ARU_MAILBOX_CAP_NEXTPTR 0h: MAILBOX_CAP_NEXTPTR_NULL 0h: MAILBOX_CAP_NEXTPTR_INIT (default)
7:0	9h	RW	T_XUSB_CFG_ARU_MAILBOX_CAP_ID: 9h: MAILBOX_CAP_ID_INIT (default)

19.13.15.2 T_XUSB_CFG_ARU_MAILBOX_CMD

Offset: 0xe4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0b00000000000000000000000000000000)

Bits	Reset	R/W	Description
31	0	R/W	T_XUSB_CFG_ARU_MAILBOX_CMD_INT_EN: 0h: MAILBOX_CMD_INT_EN_INIT (default)
30	0	RW	T_XUSB_CFG_ARU_MAILBOX_CMD_DEST_XHCI: 0h: MAILBOX_CMD_DEST_XHCI_INIT (default)
29	0	RW	T_XUSB_CFG_ARU_MAILBOX_CMD_DEST_SMI: 0h: MAILBOX_CMD_DEST_SMI_INIT (default)
28	0	RW	T_XUSB_CFG_ARU_MAILBOX_CMD_DEST_PME: 0h: MAILBOX_CMD_DEST_PME_INIT (default)
27	0	RW	T_XUSB_CFG_ARU_MAILBOX_CMD_DEST_FALCON: 0h: MAILBOX_CMD_DEST_FALCON_INIT (default)
26:0	0	RW	T_XUSB_CFG_ARU_MAILBOX_CMD_RSVD: 0h: MAILBOX_CMD_RSVD_INIT (default)

19.13.15.3 T_XUSB_CFG_ARU_MAILBOX_DATA_IN

Offset: 0xe8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0b00000000000000000000000000000000)

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_CFG_ARU_MAILBOX_DATA_IN_MSG: 0h: MAILBOX_DATA_IN_MSG_INIT (default)

19.13.15.4 T_XUSB_CFG_ARU_MAILBOX_DATA_OUT

Offset: 0xec | Read/Write: R/W | Reset: 0xXXXXXXXX (0b00000000000000000000000000000000)

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_CFG_ARU_MAILBOX_DATA_OUT_MSG: 0h: MAILBOX_DATA_OUT_MSG_INIT (default)

19.13.15.5 T_XUSB_CFG_ARU_MAILBOX_OWNER

Offset: 0xf0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0b00000000000000000000000000000000)

Bits	Reset	R/W	Description
31:8	0	R/W	T_XUSB_CFG_ARU_MAILBOX_ OWNER_RSVD: 0h: MAILBOX_OWNER_RSVD_INIT (default)
7:0	0	RW	T_XUSB_CFG_ARU_MAILBOX_ OWNER_ID: 0h: MAILBOX_OWNER_ID_INIT (default)

19.13.16 XUSB CSB Registers

19.13.16.1 XUSB_CSB_MEMPOOL_ILOAD_ATTR_0

L2IMEMOP Static Configuration Register

Offset: 0x0101a00 | Read/Write: R/W | Reset: 0b000xxxxxxxxx00000000000000000000

Bit	R/W	Reset	Description
31	RW	0x0	TC 0: TC_DEFAULT
30	RW	0x0	NS 0: NS_DEFAULT
29	RW	0x0	RO 0: RO_DEFAULT
19:8	RW	0x0	SIZE 0: SIZE_DEFAULT
7:0	RO	0x0	RSVD 0: RSVD_DEFAULT

19.13.16.2 XUSB_CSB_MEMPOOL_ILOAD_BASE_LO_0

L2IMEMOP Static Configuration Register

Offset: 0x0101a04 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	R/W	Reset	Description
31:8	RW	0x0	RSVD 0: RSVD_DEFAULT
7:0	RO	0x0	SRC_ADDR 0: SRC_ADDR_DEFAULT

19.13.16.3 XUSB_CSB_MEMPOOL_ILOAD_BASE_HI_0

L2IMEMOP Static Configuration Register

Offset: 0x0101a08 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
7:0	0x0	SRC_ADDR 0: SRC_ADDR_DEFAULT

19.13.16.4 XUSB_CSB_MEMPOOL_L2IMEMOP_SIZE_0

L2IMEMOP Operational Register

Offset: 0x0101a10 | Read/Write: R/W | Reset: 0b0000000xxxxx00000000000000000000

Bit	R/W	Reset	Description
31:24	RW	0x0	SRC_COUNT 0: SRC_COUNT_DEFAULT
19:8	RW	0x0	SRC_OFFSET 0: SRC_OFFSET_DEFAULT
7:0	RO	0x0	RSVD

Bit	R/W	Reset	Description
			0: RSVD_DEFAULT

19.13.16.5 XUSB_CSB_MEMPOOL_L2IMEMOP_TRIG_0

L2IMEMOP Operational Register

L2IMEMop Action encodings

bit[0] : 1=> RESULT, 0=> NO RESULT

bit[1] : 1=> OPTIMIZED, 0=> UNOPTIMIZED

bit[7:4] 0001 => LOAD_LOCKED

0010 => UNLOCK

0100 => INVALIDATE_ALL

0101 => QUERY_INDEX

0110 => QUERY_SPACE

Offset: 0x0101a14 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxx000000000000000000

Bit	R/W	Reset	Description
31:24	RW	X	ACTION 0x10: L2IMEM_LOAD_LOCKED 0x11: L2IMEM_LOAD_LOCKED_RESULT 0x20: L2IMEM_UNLOCK 0x21: L2IMEM_UNLOCK_RESULT 0x22: L2IMEM_UNLOCK_OPTIMIZED 0x23: L2IMEM_UNLOCK_OPTIMIZED_RESULT 0x40: L2IMEM_INVALIDATE_ALL 0x41: L2IMEM_INVALIDATE_ALL_RESULT 0x50: L2IMEM_QUERY_INDEX_RESULT 0x60: L2IMEM_QUERY_SPACE_RESULT 0xFF: WAITFENCE_RESULT
17:8	RW	0x0	DEST_INDEX 0: DEST_INDEX_DEFAULT
7:0	RO	0x0	RSVD 0: RSVD_DEFAULT

19.13.16.6 XUSB_CSB_MEMPOOL_APMAP_0

Aperture Programming Register

Offset: 0x010181c | Read/Write: R/W | Reset: 0b1xxxxxxxxxxx0xxxxxxxxxxxxxx00xxxxx000

Bit	Reset	Description
31	0x1	BOOTPATH 1: BOOTPATH_DEFAULT
24	0x0	XREQ_READ 0: XREQ_READ_DEFAULT
9:8	0x0	XMAP 0: XMAP_A 1: XMAP_B 2: XMAP_C

Bit	Reset	Description
2:0	0x0	FDDMA 0: FDDMA_A 1: FDDMA_B 2: FDDMA_C 3: FDDMA_D 4: FDDMA_E

19.13.17 XUSB Falcon Registers

19.13.17.1 FALCON_CPUCTL_0

Offset: 0x100 | Read/Write: R/W | Reset: 0b00000000000000000000000000xxxxx

Bit	Reset	R/W	Description
5	X	RO	STOPPED: This bit indicates whether the CPU is currently in the stopped state. The Falcon processor will exit this state if a 1 is written to the STARTCPU bit, or if an interrupt arrives on one of its two inputs and the corresponding IE bit in CSW is set 0: STOPPED_FALSE 1: STOPPED_TRUE
4	X	RO	HALTED: This bit indicates whether the CPU is currently in the halted state. The Falcon processor can only exit this state when a 1 is written to the STARTCPU bit. 0: HALTED_FALSE 1: HALTED_TRUE
3	X	WO	HRESET: Setting HRESET to true will apply a hard reset. This bit will auto-clear and setting to false has no effect. 0: HRESET_FALSE 1: HRESET_TRUE
2	X	WO	SRESET: Setting SRESET to true will apply a soft reset. This bit will auto-clear and setting to false has no effect. 0: SRESET_FALSE 1: SRESET_TRUE
1	X	WO	STARTCPU: Setting STARTCPU to true will start CPU execution while in a HALTED state. If a start request is still pending, setting to false cancel the start request. Writing any value has no effect while the CPU is running. 0: STARTCPU_FALSE 1: STARTCPU_TRUE
0	X	WO	IINVAL: Setting IINVAL to true causes all blocks in IMEM except block 0 to be marked as INVALID. This bit will auto-clear and setting to false has no effect. 0: IINVAL_FALSE 1: IINVAL_TRUE

19.13.17.2 FALCON_BOOTVEC_0

The BOOTVEC register stores the initial execution start address of the CPU when it is first started after a reset.

Offset: 0x104 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	VEC 0: VEC_INIT

19.13.17.3 FALCON_DMACTL_0

Offset: 0x10c | Read/Write: R/W | Reset: 0b00000000000000000000000xxxxxxx1

Bit	Reset	R/W	Description
7	X	RO	SECURE_STAT
6:3	X	RO	DMAQ_NUM: Indicates the valid request number at the DMA request queue
2	X	RO	IMEM_SCRUBBING:

Bit	Reset	R/W	Description
			0: Indicates scrubbing is done. For a non-secure Falcon, this value is always 0. 1: Indicates secure scrubber is pending and scrubbing IMEM, any access to IMEM will be blocked until scrubbing is done 0: IMEM_SCRUBBING_DONE 1: IMEM_SCRUBBING_PENDING
1	X	RO	DMEM_SCRUBBING: 0: Indicates scrubbing is done. For a non-secure Falcon, this value is always 0. 1: Indicates secure scrubber is pending and scrubbing DMEM, any access to DMEM will be blocked until scrubbing is done 0: DMEM_SCRUBBING_DONE 1: DMEM_SCRUBBING_PENDING
0	0x1	RW	REQUIRE_CTX: When REQUIRE_CTX is set to true, a valid context must be loaded before any DMA request can be serviced. Pending requests without a valid current context remain pending, and do not prevent the engine from reporting idle. When this bit is set to false, DMA requests are serviced regardless of the current context. Note that once a request is issued, it must complete before the engine will be able to report idle, as needed for example to process WFI context switch requests. 0:REQUIRE_CTX_FALSE 1: REQUIRE_CTX_TRUE 1: REQUIRE_CTX_INIT

19.13.17.4 FALCON_IMFILLRNG1_0

IMFILLRNG1 indicates tag values for the low and high end of the PC range to be auto-filled. The PC range is [tag_lo<<8 .. tag_hi<<8+255].

If the user enables the auto-fill feature and leaves IMFILLRNG1 as zero, instruction 0x0~0x0ff always works as auto-fill range.

Offset: 0x154 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	TAG_HI 0: TAG_HI_INIT
15:0	0x0	TAG_LO TAG_LO_INIT

19.13.17.5 FALCON_IMFILLCTL_0

Offset: 0x158 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
7:0	0x0	NBLOCKS 0: NBLOCKS_INIT

19.13.18 XUSB XHCI Registers

19.13.18.1 T_XUSB_XHCI_CAP_REG0

Offset: 0x00 | Read/Write: R

Bits	Reset	R/W	Description
31:16	Unknown	R	T_XUSB_XHCI_CAP_REG0_HCIVERSION: 100h: HCIVERSION_INIT
15:8	Unknown	R	T_XUSB_XHCI_CAP_REG0_RSVD: 0h: RSVD_00
7:0	Unknown	R	T_XUSB_XHCI_CAP_REG0_CAPLENGTH: 20h: CAPLENGTH_INIT

19.13.18.2 T_XUSB_XHCI_CAP_HCSPARAMS1

Offset: 0x04 | Read/Write: R

Bits	Reset	R/W	Description
31:24	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS1_MAXPORTS: 10h: MAXPORTS_INIT
23	0	R	Reserved
22:19	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS1_RSVD0: 0h: RSVD0_00
18:8	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS1_MAXINTRS: 1h: MAXINTRS_INIT
7:0	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS1_MAXSLOTS: FFh: MAXSLOTS_INIT

19.13.18.3 T_XUSB_XHCI_CAP_HCSPARAMS2

Offset: 0x08 | Read/Write: R

Bits	Reset	R/W	Description
31:27	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS2_MAXSPBLO: 0h: MAXSPBLO_INIT
26	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS2_SPR: 1h: SPR_TRUE 0h: SPR_FALSE
25:21	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS2_MAXSPBHI: 0h: MAXSPBHI_INIT
20:8	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS2_RSVD: 0h: RSVD_00
7:4	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS2_ERST_MAX: Fh: ERST_MAX_INIT
3:0	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS2_IST: 8h: IST_INIT

19.13.18.4 T_XUSB_XHCI_CAP_HCSPARAMS3

Offset: 0x0c | Read/Write: R

Bits	Reset	R/W	Description
31:16	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS3_U2LAT: 40h: U2LAT_INIT
15:8	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS3_RSVD: 0h: RSVD_00
7:0	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS3_U1LAT: 2h: U1LAT_INIT

19.13.18.5 T_XUSB_XHCI_CAP_HCCPARAMS

Offset: 0x10 | Read/Write: R

Bits	Reset	R/W	Description
31:16	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_XECP: 180h: XECP_USBLEGSUP 184h: XECP_SUPPROT_USB3 187h: XECP_SUPPROT_USB2

Bits	Reset	R/W	Description
15:12	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_MAXPSASIZE: Fh: MAXPSASIZE_INIT
11:9	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_RSVD: 0h: RSVD_00
8	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_PAE: 1h: PAE_TRUE 0h: PAE_FALSE
7	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_NSS: 1h: NSS_TRUE 0h: NSS_FALSE
6	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_LTC: 1h: LTC_TRUE 0h: LTC_FALSE
5	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_LHRC: 1h: LHRC_TRUE 0h: LHRC_FALSE
4	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_PIND: 1h: PIND_TRUE 0h: PIND_FALSE
3	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_PPC: 1h: PPC_TRUE 0h: PPC_FALSE
2	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_CSZ: 1h: CSZ_64B 0h: CSZ_32B
1	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_BNC: 1h: BNC_TRUE 0h: BNC_FALSE
0	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_AC64: 1h: AC64_TRUE 0h: AC64_FALSE

19.13.18.6 T_XUSB_XHCI_CAP_DBOFF

Offset: 0x14 | Read/Write: R

Bits	Reset	R/W	Description
31:2	Unknown	R	T_XUSB_XHCI_CAP_DBOFF_OFFSET: 300h: OFFSET_INIT
1:0	Unknown	R	T_XUSB_XHCI_CAP_DBOFF_RSVD: 0h: RSVD_00

19.13.18.7 T_XUSB_XHCI_CAP_RTSOFF

Offset: 0x18 | Read/Write: R

Bits	Reset	R/W	Description
31:5	Unknown	R	T_XUSB_XHCI_CAP_RTSOFF_OFFSET: 40h: OFFSET_INIT
4:0	Unknown	R	T_XUSB_XHCI_CAP_RTSOFF_RSVD: 0h: RSVD_00

19.13.18.8 T_XUSB_XHCI_CAP_RSVD0

Offset: 0x1c | Read/Write: R

Bits	Reset	R/W	Description
31:0	Unknown	R	T_XUSB_XHCI_CAP_RSVD0_F: 0h: F_00

19.13.18.9 T_XUSB_XHCI_OP_USBCMD

Offset: 0x20 | Read/Write: R/W

Bits	Reset	R/W	Description
31:12	0	R	T_XUSB_XHCI_OP_USBCMD_RSVD1: 0h: RSVD1_00 (default)
11	0	R/W	T_XUSB_XHCI_OP_USBCMD_EU3S: 0h: EU3S_DISABLE (default) 1h: EU3S_ENABLE
10	0	R/W	T_XUSB_XHCI_OP_USBCMD_EWE: 0h: EWE_DISABLE (default) 1h: EWE_ENABLE
9	0	RW1C	T_XUSB_XHCI_OP_USBCMD_CRS: 0h: CRS_INIT (default) 1h: CRS_START 0h: CRS_NOOP
8	0	RW1C	T_XUSB_XHCI_OP_USBCMD_CSS: 0h: CSS_INIT (default) 1h: CSS_START 0h: CSS_NOOP
7	0	R/W	T_XUSB_XHCI_OP_USBCMD_LHCRST: 0h: LHCRST_NOT_PENDING (default) 1h: LHCRST_PENDING 1h: LHCRST_SET
6:4	0	R	T_XUSB_XHCI_OP_USBCMD_RSVD0: 0h: RSVD0_00 (default)
3	0	R/W	T_XUSB_XHCI_OP_USBCMD_HSEE: 0h: HSEE_DISABLE (default) 1h: HSEE_ENABLE
2	0	R/W	T_XUSB_XHCI_OP_USBCMD_INTE: 0h: INTE_DISABLE (default) 1h: INTE_ENABLE
1	0	R/W	T_XUSB_XHCI_OP_USBCMD_HCRST: 0h: HCRST_NOT_PENDING (default) 1h: HCRST_PENDING 1h: HCRST_SET
0	0	R/W	T_XUSB_XHCI_OP_USBCMD_RS: 0h: RS_STOP (default) 1h: RS_RUN

19.13.18.10 T_XUSB_XHCI_OP_USBSTS

Offset: 0x24 | Read/Write: R/W

Bits	Reset	R/W	Description
31:13	0	R	T_XUSB_XHCI_OP_USBSTS_RSVD2: 0h: RSVD2_00 (default)
12	0	R	T_XUSB_XHCI_OP_USBSTS_HCE: 0h: HCE_NO_ERROR (default) 1h: HCE_ERROR
11	1	R	T_XUSB_XHCI_OP_USBSTS_CNR:

Bits	Reset	R/W	Description
			1h: CNR_NOT_READY (default) 0h: CNR_READY
10	0	RW1C	T_XUSB_XHCI_OP_USBSTS_SRE: 0h: SRE_NOT_PENDING (default) 1h: SRE_PENDING 1h: SRE_CLEAR
9	0	R	T_XUSB_XHCI_OP_USBSTS_RSS: 0h: RSS_NOT_PENDING (default) 1h: RSS_PENDING
8	0	R	T_XUSB_XHCI_OP_USBSTS_SSS: 0h: SSS_NOT_PENDING (default) 1h: SSS_PENDING
7:5	0	R	T_XUSB_XHCI_OP_USBSTS_RSVD1: 0h: RSVD1_00 (default)
4	0	RW1C	T_XUSB_XHCI_OP_USBSTS_PCD: 0h: PCD_NOT_PENDING (default) 1h: PCD_PENDING 1h: PCD_CLEAR
3	0	RW1C	T_XUSB_XHCI_OP_USBSTS_EINT: 0h: EINT_NOT_PENDING (default) 1h: EINT_PENDING 1h: EINT_CLEAR
2	0	RW1C	T_XUSB_XHCI_OP_USBSTS_HSE: 0h: HSE_NOT_PENDING (default) 1h: HSE_PENDING 1h: HSE_CLEAR
1	0	R	T_XUSB_XHCI_OP_USBSTS_RSVD0: 0h: RSVD0_0 (default)
0	1	R	T_XUSB_XHCI_OP_USBSTS_HCH: 1h: HCH_HALTED (default) 0h: HCH_RUNNING

19.13.18.11 T_XUSB_XHCI_OP_PGSZ

Offset: 0x28 | Read/Write: R

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_XHCI_OP_PGSZ_RSVD0: 0h: RSVD0_00 (default)
15:0	1	R	T_XUSB_XHCI_OP_PGSZ_PAGESIZE: 1h: PAGESIZE_4K (default)

19.13.18.12 T_XUSB_XHCI_OP_DNCTRL

Offset: 0x34 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	Reserved
15	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N15: 0h: N15_DISABLE (default) 1h: N15_ENABLE
14	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N14: 0h: N14_DISABLE (default) 1h: N14_ENABLE

Bits	Reset	R/W	Description
13	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N13: 0h: N13_DISABLE (default) 1h: N13_ENABLE
12	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N12: 0h: N12_DISABLE (default) 1h: N12_ENABLE
11	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N11: 0h: N11_DISABLE (default) 1h: N11_ENABLE
10	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N10: 0h: N10_DISABLE (default) 1h: N10_ENABLE
9	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N9: 0h: N9_DISABLE (default) 1h: N9_ENABLE
8	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N8: 0h: N8_DISABLE (default) 1h: N8_ENABLE
7	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N7: 0h: N7_DISABLE (default) 1h: N7_ENABLE
6	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N6: 0h: N6_DISABLE (default) 1h: N6_ENABLE
5	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N5: 0h: N5_DISABLE (default) 1h: N5_ENABLE
4	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N4: 0h: N4_DISABLE (default) 1h: N4_ENABLE
3	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N3: 0h: N3_DISABLE (default) 1h: N3_ENABLE
2	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N2: 0h: N2_DISABLE (default) 1h: N2_ENABLE
1	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N1: 0h: N1_DISABLE (default) 1h: N1_ENABLE
0	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N0: 0h: N0_DISABLE (default) 1h: N0_ENABLE

19.13.18.13 T_XUSB_XHCI_OP_CRCR0

Offset: 0x38 | Read/Write: R/W

Bits	Reset	R/W	Description
31:6	0	W	T_XUSB_XHCI_OP_CRCR0_CRPLO: 0h: CRPLO_INIT (default)
5:4	0	R	T_XUSB_XHCI_OP_CRCR0_RSVD0: 0h: RSVD0_00 (default)
3	0	R	T_XUSB_XHCI_OP_CRCR0_CRR: 0h: CRR_STOPPED (default) 1h: CRR_RUNNING

Bits	Reset	R/W	Description
2	0	RW1C	T_XUSB_XHCI_OP_CRCR0_CA: 0h: CA_INIT (default) 1h: CA_ABORT
1	0	RW1C	T_XUSB_XHCI_OP_CRCR0_CS: 0h: CS_INIT (default) 1h: CS_STOP
0	0	RW1C	T_XUSB_XHCI_OP_CRCR0_RCS: 0h: RCS_0 (default) 1h: RCS_1

19.13.18.14 T_XUSB_XHCI_OP_CRCR1

Offset: 0x3c | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	W	T_XUSB_XHCI_OP_CRCR1_CRPHI: 0h: CRPHI_INIT (default)

19.13.18.15 T_XUSB_XHCI_OP_DCBAAP0

Offset: 0x50 | Read/Write: R/W

Bits	Reset	R/W	Description
31:6	0	R/W	T_XUSB_XHCI_OP_DCBAAP0_DCBAAPLO: 0h: DCBAAPLO_INIT (default)
5:0	0	R	T_XUSB_XHCI_OP_DCBAAP0_RSVD0: 0h: RSVD0_00 (default)

19.13.18.16 T_XUSB_XHCI_OP_DCBAAP1

Offset: 0x54 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_XHCI_OP_DCBAAP1_DCBAAPHI: 0h: DCBAAPHI_INIT (default)

19.13.18.17 T_XUSB_XHCI_OP_CONFIG

Offset: 0x58 | Read/Write: R/W

Bits	Reset	R/W	Description
31:8	0	R	T_XUSB_XHCI_OP_CONFIG_RSVD0: 0h: RSVD0_00 (default)
7:0	0	R/W	T_XUSB_XHCI_OP_CONFIG_MAXSLOTSEN: 0h: MAXSLOTSEN_INIT (default)

19.13.18.18 T_XUSB_XHCI_OP_PORTSC

Offset: 0x420 – 0x510 | Read/Write: R/W

Bits	Reset	R/W	Description
31	0	R	T_XUSB_XHCI_OP_PORTSC_WPR: 0h: WPR_NOT_PENDING (default) 1h: WPR_PENDING 1h: WPR_SET

Bits	Reset	R/W	Description
30	0	R	T_XUSB_XHCI_OP_PORTSC_DR: 0h: DR_FALSE (default) 1h: DR_TRUE
29:28	0	R	T_XUSB_XHCI_OP_PORTSC_RSVD2: 0h: RSVD2_00 (default)
27	0	R/W	T_XUSB_XHCI_OP_PORTSC_WOE: 0h: WOE_DISABLED (default) 1h: WOE_ENABLED
26	0	R/W	T_XUSB_XHCI_OP_PORTSC_WDE: 0h: WDE_DISABLED (default) 1h: WDE_ENABLED
25	0	R/W	T_XUSB_XHCI_OP_PORTSC_WCE: 0h: WCE_DISABLED (default) 1h: WCE_ENABLED
24	0	R	T_XUSB_XHCI_OP_PORTSC_CAS: 0h: CAS_INIT (default)
23	0	RW1C	T_XUSB_XHCI_OP_PORTSC_CEC: 0h: CEC_NOT_PENDING (default) 1h: CEC_PENDING 1h: CEC_CLEAR
22	0	RW1C	T_XUSB_XHCI_OP_PORTSC_PLC: 0h: PLC_NOT_PENDING (default) 1h: PLC_PENDING 1h: PLC_CLEAR
21	0	RW1C	T_XUSB_XHCI_OP_PORTSC_PRC: 0h: PRC_NOT_PENDING (default) 1h: PRC_PENDING 1h: PRC_CLEAR
20	0	RW1C	T_XUSB_XHCI_OP_PORTSC_OCC: 0h: OCC_NOT_PENDING (default) 1h: OCC_PENDING 1h: OCC_CLEAR
19	0	RW1C	T_XUSB_XHCI_OP_PORTSC_WRC: 0h: WRC_NOT_PENDING (default) 1h: WRC_PENDING 1h: WRC_CLEAR
18	0	RW1C	T_XUSB_XHCI_OP_PORTSC_PEC: 0h: PEC_NOT_PENDING (default) 1h: PEC_PENDING 1h: PEC_CLEAR
17	0	RW1C	T_XUSB_XHCI_OP_PORTSC_CSC: 0h: CSC_NOT_PENDING (default) 1h: CSC_PENDING 1h: CSC_CLEAR
16	0	RW1C	T_XUSB_XHCI_OP_PORTSC_LWS: 0h: LWS_DISABLED (default) 1h: LWS_ENABLED
15:14	0	R/W	T_XUSB_XHCI_OP_PORTSC_PIC: 0h: PIC_OFF (default) 1h: PIC_AMBER 2h: PIC_GREEN 3h: PIC_UNDEFINED
13:10	0	R	T_XUSB_XHCI_OP_PORTSC_PSPD: 0h: PSPD_UNDEFINED (default) 1h: PSPD_FS 2h: PSPD_LS 3h: PSPD_HS

Bits	Reset	R/W	Description
			4h: PSPD_SS Fh: PSPD_UNKNOWN
9	1	R/W	T_XUSB_XHCI_OP_PORTSC_PP: 0h: PP_OFF 1h: PP_ON (default)
8:5	4h	RW1C	T_XUSB_XHCI_OP_PORTSC_PLS: 0h: PLS_U0 1h: PLS_U1 2h: PLS_U2 3h: PLS_U3 4h: PLS_DISABLED (default) 5h: PLS_RXDETECT 6h: PLS_INACTIVE 7h: PLS_POLLING 8h: PLS_RECOVERY 9h: PLS_HOTRESET Ah: PLS_COMPLIANCE Bh: PLS_LOOPBACK Fh: PLS_RESUME 0h: PLS_GOTO_U0 1h: PLS_GOTO_U1 2h: PLS_GOTO_U2 3h: PLS_GOTO_U3
4	0	R/W	T_XUSB_XHCI_OP_PORTSC_PR: 0h: PR_NOT_PENDING (default) 1h: PR_PENDING 1h: PR_SET
3	0	R	T_XUSB_XHCI_OP_PORTSC_OCA: 0h: OCA_FALSE (default) 1h: OCA_TRUE
2	0	R	T_XUSB_XHCI_OP_PORTSC_RSVD0: 0h: RSVD0_0 (default)
1	0	RW1C	T_XUSB_XHCI_OP_PORTSC_PED: 0h: PED_DISABLED (default) 1h: PED_ENABLED 1h: PED_CLEAR
0	0	R	T_XUSB_XHCI_OP_PORTSC_CCS: 0h: CCS_NODEV (default) 1h: CCS_DEV

19.13.18.19 T_XUSB_XHCI_OP_PORTPMSCSS

Offset: 0x424 – 0x514 | Read/Write: R/W

Bits	Reset	R/W	Description
31:17	0	R	T_XUSB_XHCI_OP_PORTPMSCSS_RSVD0: 0h: RSVD0_00 (default)
16	0	R/W	T_XUSB_XHCI_OP_PORTPMSCSS_FLA: 0h: FLA_INIT (default)
15:8	0	R/W	T_XUSB_XHCI_OP_PORTPMSCSS_U2TIMEOUT: 0h: U2TIMEOUT_INIT (default)
7:0	0	R/W	T_XUSB_XHCI_OP_PORTPMSCSS_U1TIMEOUT: 0h: U1TIMEOUT_INIT (default)

19.13.18.20 T_XUSB_XHCI_OP_PORTPMSCHS

Offset: 0x424 – 0x514 | Read/Write: R/W

Bits	Reset	R/W	Description
31:28	0	R/W	T_XUSB_XHCI_OP_PORTPMSCHS_TM: 0h: TM_DISABLE (default) 1h: TM_JSTATE 2h: TM_KSTATE 3h: TM_SE0NAK 4h: TM_PACKET 5h: TM_FORCEEN Fh: TM_ERROR
27:17	0	R	T_XUSB_XHCI_OP_PORTPMSCHS_RSVD0: 0h: RSVD0_00 (default)
16	0	R/W	T_XUSB_XHCI_OP_PORTPMSCHS_HLE: 0h: HLE_INIT (default)
15:8	0	R/W	T_XUSB_XHCI_OP_PORTPMSCHS_L1DS: 0h: L1DS_INIT (default)
7:4	0	R/W	T_XUSB_XHCI_OP_PORTPMSCHS_HIRD: 0h: HIRD_INIT (default)
3	0	R/W	T_XUSB_XHCI_OP_PORTPMSCHS_RWE: 0h: RWE_DISABLED (default) 1h: RWE_ENABLED
2:0	0	R	T_XUSB_XHCI_OP_PORTPMSCHS_L1S: 0h: L1S_INVLD (default) 1h: L1S_SUCCESS 2h: L1S_NYET 3h: L1S_STALL 4h: L1S_ERROR

19.13.18.21 T_XUSB_XHCI_OP_PORTLISC

Offset: 0x428 – 0x458 | Read/Write: R

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_XHCI_OP_PORTLISC_RSVD0: 0h: RSVD0_00 (default)
15:0	0	R	T_XUSB_XHCI_OP_PORTLISC_LEC: 0h: LEC_INIT (default)

19.13.18.22 T_XUSB_XHCI_OP_PORTHLPMC

Offset: 0x42c – 0x51c | Read/Write: R

Bits	Reset	R/W	Description
13:10	0	R	T_XUSB_XHCI_OP_PORTHLPMC_BESLD: 0h: BESLD_INIT (default)
9:2	0	R	T_XUSB_XHCI_OP_ PORTHLPMC_L1_TIMEOUT: 0h: L1_TIMEOUT_INIT (default)
1:0	0	R	T_XUSB_XHCI_OP_ PORTHLPMC_HIRDM: 0h: HIRDM_INIT (default)

19.13.18.23 T_XUSB_XHCI_EC_USBLEGSUP

Offset: 0x600 | Read/Write: R/W

Bits	Reset	R/W	Description
31:25	0	R	T_XUSB_XHCI_EC_USBLEGSUP_RSVD1: 0h: RSVD1_00 (default)

Bits	Reset	R/W	Description
24	0	R/W	T_XUSB_XHCI_EC_USBLEGSUP_OSSEM: 0h: OSSEM_INIT (default)
23:17	0	R	T_XUSB_XHCI_EC_USBLEGSUP_RSVD0: 0h: RSVD0_00 (default)
16	0	R/W	T_XUSB_XHCI_EC_USBLEGSUP_BIOSSEM: 0h: BIOSSEM_INIT (default)
15:8	4h	R	T_XUSB_XHCI_EC_USBLEGSUP_NEXT: 4h: NEXT_SUPPROT_USB3 (default)
7:0	1h	R	T_XUSB_XHCI_EC_USBLEGSUP_CAPID: 1h: CAPID_USBLEGSUP (default)

19.13.18.24 T_XUSB_XHCI_EC_USBLEGCTLSTS

Offset: 0x604 | Read/Write: R/W

Bits	Reset	R/W	Description
31	0	RW1C	T_XUSB_XHCI_EC_USBLEGCTLSTS_BAR: 0h: BAR_NOT_PENDING (default) 1h: BAR_PENDING 1h: BAR_CLEAR
30	0	RW1C	T_XUSB_XHCI_EC_USBLEGCTLSTS_PCIC: 0h: PCIC_NOT_PENDING (default) 1h: PCIC_PENDING 1h: PCIC_CLEAR
29	0	RW1C	T_XUSB_XHCI_EC_USBLEGCTLSTS_OSOC: 0h: OSOC_NOT_PENDING (default) 1h: OSOC_PENDING 1h: OSOC_CLEAR
28:21	0	R	T_XUSB_XHCI_EC_USBLEGCTLSTS_RSVD3: 0h: RSVD3_00 (default)
20	0	R	T_XUSB_XHCI_EC_USBLEGCTLSTS_HSE: 0h: HSE_NOT_PENDING (default) 1h: HSE_PENDING
19:17	0	R	T_XUSB_XHCI_EC_USBLEGCTLSTS_RSVD2: 0h: RSVD2_00 (default)
16	0	R	T_XUSB_XHCI_EC_USBLEGCTLSTS_EVI: 0h: EVI_NOT_PENDING (default) 1h: EVI_PENDING
15	0	R/W	T_XUSB_XHCI_EC_USBLEGCTLSTS_BAREN: 0h: BAREN_DISABLED (default) 1h: BAREN_ENABLED
14	0	R/W	T_XUSB_XHCI_EC_USBLEGCTLSTS_PCIEN: 0h: PCIEN_DISABLED (default) 1h: PCIEN_ENABLED
13	0	R/W	T_XUSB_XHCI_EC_USBLEGCTLSTS_OSOEN: 0h: OSOEN_DISABLED (default) 1h: OSOEN_ENABLED
12:5	0	R	T_XUSB_XHCI_EC_USBLEGCTLSTS_RSVD1: 0h: RSVD1_00 (default)
4	0	R/W	T_XUSB_XHCI_EC_USBLEGCTLSTS_HSEEN: 0h: HSEEN_DISABLED (default) 1h: HSEEN_ENABLED
3:1	0	R	T_XUSB_XHCI_EC_USBLEGCTLSTS_RSVD0: 0h: RSVD0_00 (default)

Bits	Reset	R/W	Description
0	0	R/W	T_XUSB_XHCI_EC_USBLEGCTLSTS_SMIEN: 0h: SMIEN_DISABLED (default) 1h: SMIEN_ENABLED

19.13.18.25 T_XUSB_XHCI_EC_SUPPROT_USB3_0

Offset: 0x610 | Read/Write: R

Bits	Reset	R/W	Description
31:24	3h	R	T_XUSB_XHCI_EC_SUPPROT_USB3_0_MAJORREV: 3h: MAJORREV_3 (default)
23:16	0	R	T_XUSB_XHCI_EC_SUPPROT_USB3_0_MINORREV: 0h: MINORREV_0 (default)
15:8	4h	R	T_XUSB_XHCI_EC_SUPPROT_USB3_0_NEXT: 4h: NEXT_SUPPROT_USB2 (default)
7:0	2h	R	T_XUSB_XHCI_EC_SUPPROT_USB3_0_CAPID: 2h: CAPID_SUPPROT_USB3 (default)

19.13.18.26 T_XUSB_XHCI_EC_SUPPROT_USB3_1

Offset: 0x614 | Read/Write: R

Bits	Reset	R/W	Description
31:0	20425355h	R	T_XUSB_XHCI_EC_SUPPROT_USB3_1_NAMESTR: 20425355h: NAMESTR_USB (default)

19.13.18.27 T_XUSB_XHCI_EC_SUPPROT_USB3_2

Offset: 0x618 | Read/Write: R

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_XHCI_EC_SUPPROT_USB3_2_RSVD0: 0h: RSVD0_00 (default)
15:8	Unknown	R	T_XUSB_XHCI_EC_SUPPROT_USB3_2_PORTCNT:
7:0	1h	R	T_XUSB_XHCI_EC_SUPPROT_USB3_2_PORTOFS: 1h: PORTOFS_VAL (default)

19.13.18.28 T_XUSB_XHCI_EC_SUPPROT_USB3_3

Offset: 0x61c | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_XHCI_EC_SUPPROT_USB3_3_SLOTTYPE: 0h: SLOTTYPE_VAL (default)

19.13.18.29 T_XUSB_XHCI_EC_SUPPROT_USB2_0

Offset: 0x620 | Read/Write: R

Bits	Reset	R/W	Description
31:24	2h	R	T_XUSB_XHCI_EC_SUPPROT_USB2_0_MAJORREV: 2h: MAJORREV_3 (default)
23:16	0h	R	T_XUSB_XHCI_EC_SUPPROT_USB2_0_MINORREV: 0h: MINORREV_0 (default)

Bits	Reset	R/W	Description
15:8	4h	R	T_XUSB_XHCI_EC_SUPPROT_USB2_0_NEXT: 4h: NEXT_DBCAP (default)
7:0	2h	R	T_XUSB_XHCI_EC_SUPPROT_USB2_0_CAPID: 2h: CAPID_SUPPROT_USB3 (default)

19.13.18.30 T_XUSB_XHCI_EC_SUPPROT_USB2_1

Offset: 0x624 | Read/Write: R

Bits	Reset	R/W	Description
31:0	20425355h	R	T_XUSB_XHCI_EC_SUPPROT_USB2_1_NAMESTR: 20425355h: NAMESTR_USB (default)

19.13.18.31 T_XUSB_XHCI_EC_SUPPROT_USB2_2

Offset: 0x628 | Read/Write: R

Bits	Reset	R/W	Description
31:28	0	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_RSVD2: 0h: RSVD2_00 (default)
27:25	0	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_MHD: 0h: MHD (default)
24:21	0	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_RSVD1: 0h: RSVD1_00 (default)
20	1	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_BLC: 1h: BLC_TRUE (default)
19	1	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_HLC: 1h: HLC_TRUE (default)
18	0	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_IHI: 0h: IHI_TRUE (default)
17	0	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_HSO: 0h: HSO_TRUE (default)
16	0	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_RSVD0: 0h: RSVD0_00 (default)
15:8	Unknown	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_PORTCNT:
7:0	Unknown	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_PORTOFS:

19.13.18.32 T_XUSB_XHCI_EC_SUPPROT_USB2_3

Offset: 0x62c | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_XHCI_EC_SUPPROT_USB2_3_SLOTTYPE: 0h: SLOTTYPE_VAL (default)

19.13.18.33 T_XUSB_XHCI_EC_DBCAP_DCID

Offset: 0x630 | Read/Write: R

Bits	Reset	R/W	Description
31:21	0	R	Reserved
20:16	1h	R	T_XUSB_XHCI_EC_DBCAP_DCID_DCERSTM:

Bits	Reset	R/W	Description
			1h: DCERSTM_VALUE (default)
15:8	0	R	T_XUSB_XHCI_EC_DBCAP_DCID_NEXT: 0h: NEXT_NONE (default)
7:0	Ah	R	T_XUSB_XHCI_EC_DBCAP_DCID_CAPID: Ah: CAPID_DBCAP (default)

19.13.18.34 T_XUSB_XHCI_EC_DBCAP_DCDB

Offset: 0x634 | Read/Write: W

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_XHCI_EC_DBCAP_DCDB_RSVD1: 0h: RSVD1_00 (default)
15:8	0	W	T_XUSB_XHCI_EC_DBCAP_DCDB_DBTARGET: 0h: DBTARGET_INIT (default)
7:0	0	R	T_XUSB_XHCI_EC_DBCAP_DCDB_RSVD0: 0h: RSVD0_00 (default)

19.13.18.35 T_XUSB_XHCI_EC_DBCAP_DCERSTSZ

Offset: 0x638 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_XHCI_EC_DBCAP_DCERSTSZ_RSVD0: 0h: RSVD0_00 (default)
15:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCERSTSZ_ERSTSZ: 0h: ERSTSZ_INIT (default)

19.13.18.36 T_XUSB_XHCI_EC_DBCAP_RSVD0

Offset: 0x63c | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_XHCI_EC_DBCAP_RSVD0_RSVD0: 0h: RSVD0_00 (default)

19.13.18.37 T_XUSB_XHCI_EC_DBCAP_DCERSTBALO

Offset: 0x640 | Read/Write: R/W

Bits	Reset	R/W	Description
31:4	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCERSTBALO_ADDRLO: 0h: ADDRLO_INIT (default)
3:0	0	R	T_XUSB_XHCI_EC_DBCAP_DCERSTBALO_RSVD0: 0h: RSVD0_00 (default)

19.13.18.38 T_XUSB_XHCI_EC_DBCAP_DCERSTBAHI

Offset: 0x644 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCERSTBAHI_ADDRHI: 0h: ADDRHI_INIT (default)

19.13.18.39 T_XUSB_XHCI_EC_DBCAP_DCERDPLO

Offset: 0x648 | Read/Write: R/W

Bits	Reset	R/W	Description
31:4	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCERDPLO_ADDRLO: 0h: ADDRLO_INIT (default)
3	0	R	T_XUSB_XHCI_EC_DBCAP_DCERDPLO_RSVD0: 0h: RSVD0_00 (default)
2:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCERDPLO_DESI: 0h: DESI_INIT (default)

19.13.18.40 T_XUSB_XHCI_EC_DBCAP_DCERDPHI

Offset: 0x64c | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCERDPHI_ADDRHI: 0h: ADDRHI_INIT (default)

19.13.18.41 T_XUSB_XHCI_EC_DBCAP_DCCTRL

Offset: 0x650 | Read/Write: R/W

Bits	Reset	R/W	Description
31	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCCTRL_DCE: 0h DCE_DIS (default) 1h DCE_EN
30:24	0	R	T_XUSB_XHCI_EC_DBCAP_DCCTRL_DEVADR: 0h: DEVADR_INIT (default)
23:16	15h	R	T_XUSB_XHCI_EC_DBCAP_DCCTRL_MAXBURST: 15h: MAXBURST_INIT (default)
15:5	0	R	T_XUSB_XHCI_EC_DBCAP_DCCTRL_RSVD0: 0h: RSVD0_00 (default)
4	0	RW1C	T_XUSB_XHCI_EC_DBCAP_DCCTRL_DRC: 0h: DRC_INIT (default) 1h: DRC_SET 1h: DRC_CLEAR
3	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCCTRL_HIT: 0h: HIT_FALSE (default) 1h: HIT_TRUE
2	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCCTRL_HOT: 0h: HOT_FALSE (default) 1h: HOT_TRUE
1	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCCTRL_LSE: 0h: LSE_DIS (default) 1h: LSE_EN
0	0	R	T_XUSB_XHCI_EC_DBCAP_DCCTRL_DCR: 0h: DCR_STOP (default) 1h: DCR_RUN

19.13.18.42 T_XUSB_XHCI_EC_DBCAP_DCST

Offset: 0x654 | Read/Write: R

Bits	Reset	R/W	Description
31:24	0	R	T_XUSB_XHCI_EC_DBCAP_DCST_DPN: 0h: DPN_INIT (default)
23:1	0	R	T_XUSB_XHCI_EC_DBCAP_DCST_RSVD: 0h: RSVD_00 (default)
0	0	R	T_XUSB_XHCI_EC_DBCAP_DCST_ER: 0h: ER_EMPTY (default) 1h: ER_NOTEMPTY

19.13.18.43 T_XUSB_XHCI_EC_DBCAP_DCPORTSC

Offset: 0x658 | Read/Write: R/W

Bits	Reset	R/W	Description
31:24	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_RSVD4: 0h: RSVD4_00 (default)
23	0	RW1C	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_CEC: 0h: CEC_NOT_PENDING (default) 1h: CEC_PENDING 1h: CEC_CLEAR
22	0	RW1C	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_PLC: 0h: PLC_NOT_PENDING (default) 1h: PLC_PENDING 1h: PLC_CLEAR
21	0	RW1C	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_PRC: 0h: PRC_NOT_PENDING (default) 1h: PRC_PENDING 1h: PRC_CLEAR
20:18	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_RSVD3: 0h: RSVD3_00 (default)
17	0	RW1C	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_CSC: 0h: CSC_NOT_PENDING (default) 1h: CSC_PENDING 1h: CSC_CLEAR
16:14	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_RSVD2: 0h: RSVD2_00 (default)
13:10	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_PS: 0h: PS_UNDEFINED (default) 4h: PS_SS
9	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_RSVD1: 0h: RSVD1_00 (default)
8:5	4h	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_PLS: 0h: PLS_U0 1h: PLS_U1 2h: PLS_U2 3h: PLS_U3 4h: PLS_DISABLED (default) 5h: PLS_RXDETECT 6h: PLS_INACTIVE 7h: PLS_POLLING 8h: PLS_RECOVERY 9h: PLS_HOTRESET Ah: PLS_COMPLIANCE Bh: PLS_LOOPBACK Fh: PLS_RESUME

Bits	Reset	R/W	Description
4	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_PR: 0h: PR_NORST (default) 1h: PR_RST
3:2	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_RSVD0: 0h: RSVD0_00 (default)
1	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_PED: 0h: PED_DIS (default) 1h: PED_EN
0	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_CCS: 0h: CCS_NOCON (default) 1h: CCS_CON

19.13.18.44 T_XUSB_XHCI_EC_DBCAP_RSVD1

Offset: 0x65c | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_XHCI_EC_DBCAP_RSVD1_RSVD0: 0h: RSVD0_00 (default)

19.13.18.45 T_XUSB_XHCI_EC_DBCAP_DCECPLO

Offset: 0x660 | Read/Write: R/W

Bits	Reset	R/W	Description
31:4	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCECPLO_ADDRLO: 0h: ADDRLO_INIT (default)
3:0	0	R	T_XUSB_XHCI_EC_DBCAP_DCECPLO_RSVD0: 0h: RSVD0_00 (default)

19.13.18.46 T_XUSB_XHCI_EC_DBCAP_DCECPHI

Offset: 0x664 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCECPHI_ADDRHI: 0h: ADDRHI_INIT (default)

19.13.18.47 T_XUSB_XHCI_EC_DBCAP_INFO0

Offset: 0x668 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R/W	T_XUSB_XHCI_EC_DBCAP_INFO0_VENDORID: 0h: VENDORID_INIT (default)
15:8	0	R	Reserved
7:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_INFO0_PROTOCOL: 0h: PROTOCOL_VENDOR (default) 1h: PROTOCOL_GNU

19.13.18.48 T_XUSB_XHCI_EC_DBCAP_INFO1

Offset: 0x66c | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R/W	T_XUSB_XHCI_EC_DBCAP_INFO1_DEV_REV: 0h: DEV_REV_INIT (default)
15:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_INFO1_PRODUCTID: 0h: PRODUCTID_INIT (default)

19.13.18.49 T_XUSB_XHCI_RT_MFINDEX

Offset: 0x800 | Read/Write: R

Bits	Reset	R/W	Description
31:14	0	R	T_XUSB_XHCI_RT_MFINDEX_RSVD0: 0h: RSVD0_00
13:0	Unknown	R	T_XUSB_XHCI_RT_MFINDEX_MFINDEX:

19.13.18.50 T_XUSB_XHCI_RT_IMAN

Offset: 0x820 | Read/Write: R/W

Bits	Reset	R/W	Description
31:2	0	R	T_XUSB_XHCI_RT_IMAN_RSVD0: 0h: RSVD0_00
1	0	R/W	T_XUSB_XHCI_RT_IMAN_IE: 0h: IE_DISABLED (default) 1h: IE_ENABLED
0	0	RW1C	T_XUSB_XHCI_RT_IMAN_IP: 0h: IP_NOT_PENDING (default) 1h: IP_PENDING 1h: IP_CLEAR

19.13.18.51 T_XUSB_XHCI_RT_IMOD

Offset: 0x824 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	Unknown	R/W	T_XUSB_XHCI_RT_IMOD_IMODC:
15:0	0FA0h	R/W	T_XUSB_XHCI_RT_IMOD_IMODI: FA0h: IMODI_INIT (default)

19.13.18.52 T_XUSB_XHCI_RT_ERSTSZ

Offset: 0x828 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_XHCI_RT_ERSTSZ_RSVD0: 0h: RSVD0_00
15:0	0	R/W	T_XUSB_XHCI_RT_ERSTSZ_SZ: 0h: SZ_INIT (default)

19.13.18.53 T_XUSB_XHCI_RT_ERRSVD

Offset: 0x82c | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_XHCI_RT_ERRSVD_RSVD0:

Bits	Reset	R/W	Description
			0h: RSVD0_00

19.13.18.54 T_XUSB_XHCI_RT_ERSTBA0

Offset: 0x830 | Read/Write: R/W

Bits	Reset	R/W	Description
31:4	0	R/W	T_XUSB_XHCI_RT_ERSTBA0_ERSTBLO: 0h: ERSTBLO_INIT (default)
3:0	0	R	T_XUSB_XHCI_RT_ERSTBA0_RSVD0: 0h: RSVD0_00

19.13.18.55 T_XUSB_XHCI_RT_ERSTBA1

Offset: 0x834 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_XHCI_RT_ERSTBA1_ERSTBHI: 0h: ERSTBHI_INIT (default)

19.13.18.56 T_XUSB_XHCI_RT_ERDP0

Offset: 0x838 | Read/Write: R/W

Bits	Reset	R/W	Description
31:4	0	R/W	T_XUSB_XHCI_RT_ERDP0_DQPTRLO: 0h: DQPTRLO_INIT (default)
2:0	0	R/W	T_XUSB_XHCI_RT_ERDP0_DESI: 0h: DESI_INIT (default)

19.13.18.57 T_XUSB_XHCI_RT_ERDP1

Offset: 0x83c | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_XHCI_RT_ERDP1_DQPTRHI: 0h: DQPTRHI_INIT (default)

19.13.18.58 T_XUSB_XHCI_DB

Offset: 0xc00 – 0xffc | Read/Write: W

Bits	Reset	R/W	Description
31:16	0	W	T_XUSB_XHCI_DB_STREAMID: 0h: STREAMID_INIT (default)
15:8	0	R	T_XUSB_XHCI_DB_RSVD0: 0h: RSVD0_00
7:0	0	W	T_XUSB_XHCI_DB_TARGET: 0h: TARGET_INIT (default)

19.13.19 USB3 UTMIP Configuration Registers

Note: Current UTMIP configuration registers use settings that support NV software drivers and should NOT be modified. If designing custom drivers, consult your NV representative prior to modifying these registers

19.13.19.1 USB3_UTMIP_PLL_CFG0_0

USB_PHY PLL Configuration Register 0

Note: This register has been moved to the Clock and Reset section of this document. PROGRAMMING THIS REGISTER HERE WILL HAVE NO EFFECT!

This register was used to configure the PLL inside UTMIP block prior to Tegra K1 processors. This has been defeatured from UTMIP space and moved to Clock and Reset (CAR) space. Refer to the UTMIP_PLL register descriptions in the Clock and Reset Controller section.

19.13.19.2 USB3_UTMIP_PLL_CFG1_0

UTMIP PLL and PLLU Configuration Register 1

Note: This register has been moved to the Clock and Reset section of this document. PROGRAMMING THIS REGISTER HERE WILL HAVE NO EFFECT!

This register was used to configure the PLL inside the UTMIP block prior to Tegra K1 processors. This has been defeatured from UTMIP space and moved to CAR space. Refer to the UTMIP_PLL register descriptions in the Clock and Reset Controller section.

19.13.19.3 USB3_UTMIP_XCVR_CFG0_0

UTMIP Transceiver Cell Configuration Register 0

Offset: 0x808 | Read/Write: R/W | Reset: 0b00100000001001010110010100000000

Bit	Reset	Description
31:25	0x10	UTMIP_XCVR_HSSLEW_MSB: Most significant bits of HS_SLEW.
24:22	0x0	UTMIP_XCVR_SETUP_MSB: Most significant bits of SETUP.
21	0x1	UTMIP_XCVR_LSBIAS_SEL: Low speed bias selection method for usb transceiver pad
20	0x0	UTMIP_XCVR_DISCON_METHOD: Disconnect method on the usb transceiver pad
19	0x0	UTMIP_FORCE_PDZI_POWERUP: Force PDZI input into power up.
18	0x1	UTMIP_FORCE_PDZI_POWERDOWN: Force PDZI input into power down. (Overrides FORCE_PDZI_POWERUP.)
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.

19.13.19.4 USB3_UTMIP_BIAS_CFG0_0

UTMIP Bias Cell Configuration Register 0

Offset: 0x80c | Read/Write: R/W | Reset: 0bx00000001100000000000110000000000

Bit	Reset	Description
30	0x0	UTMIP_IDDIG_C_VAL: See IDDIG_C_SEL.
29	0x0	UTMIP_IDDIG_C_SEL: 0: IdDig_c = IdDig_c. 1: IdDig_c = IDDIG_C_VAL.
28	0x0	UTMIP_IDDIG_B_VAL: See IDDIG_B_SEL.
27	0x0	UTMIP_IDDIG_B_SEL: 0: IdDig_b = IdDig_b. 1: IdDig_b = IDDIG_B_VAL.
26	0x0	UTMIP_IDDIG_A_VAL: See IDDIG_A_SEL.
25	0x0	UTMIP_IDDIG_A_SEL: 0: IdDig_a = IdDig_a. 1: IdDig_a = IDDIG_A_VAL.
24	0x0	UTMIP_HSDISCON_LEVEL_MSB: Most significant bit of UTMIP_HSDISCON_LEVEL, bit 2
23	0x1	UTMIP_IDPD_VAL: See IDPD_SEL.
22	0x1	UTMIP_IDPD_SEL: 0: Reserved. Refer to the PMC registers for this feature.
21	0x0	UTMIP_IDDIG_VAL: See IDDIG_SEL.
20	0x0	UTMIP_IDDIG_SEL: 0: IdDig = IdDig. 1: IdDig = IDDIG_VAL.
19	0x0	UTMIP_GPI_VAL: See GPI_SEL.
18	0x0	UTMIP_GPI_SEL: 0: StaticGpi = IdDig. 1: StaticGpi = GPI_VAL.
17:15	0x0	UTMIP_ACTIVE_TERM_OFFSET: Active termination control offset.
14:12	0x0	UTMIP_ACTIVE_PULLUP_OFFSET: Active 1.5K pull-up control offset.
11	0x1	UTMIP_OTGPD: Power down OTG circuit.
10	0x1	UTMIP_BIASPD: Power down bias circuit.
9:8	0x0	UTMIP_VBUS_LEVEL_LEVEL: Vbus detector level.
7:6	0x0	UTMIP_SESS_LEVEL_LEVEL: SessionEnd detector level.
5:4	0x0	UTMIP_HSCHIRP_LEVEL: HS chirp detector level.
3:2	0x0	UTMIP_HSDISCON_LEVEL: HS disconnect detector level.
1:0	0x0	UTMIP_HSSQUELCH_LEVEL: HS squelch detector level.

19.13.19.5 USB3_UTMIP_HSRX_CFG0_0

UTMIP High Speed Receive Config 0

Offset: 0x810 | Read/Write: R/W | Reset: 0b10010001011001010011010000000000

Bit	Reset	Description
31:30	0x2	UTMIP_KEEP_PATT_ON_ACTIVE: Keep the stay alive pattern on active
29	0x0	UTMIP_ALLOW_CONSEC_UPDN: Allow consecutive ups and downs on the bits, debug only, set to 0.
28	0x1	UTMIP_REALIGN_ON_NEW_PKT: Realign the inertia counters on a new packet
27:24	0x1	UTMIP_PCOUNT_UPDN_DIV: The number of (edges-1) needed to move the sampling point
23:21	0x3	UTMIP_SQUELCH_EOP_DLY: Limit the delay of the squelch at EOP time
20	0x0	UTMIP_NO_STRIPPING: Do not strip incoming data
19:15	0xa	UTMIP_IDLE_WAIT: Number of idle cycles to declare IDLE.

Bit	Reset	Description
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

19.13.19.6 USB3_UTMIP_HSRX_CFG1_0

UTMIP High Speed Receive Config 1

Offset: 0x814 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxx010011

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

19.13.19.7 USB3_UTMIP_FSLSRX_CFG0_0

UTMIP Full and Low Speed Receive Config 0

Offset: 0x818 | Read/Write: R/W | Reset: 0b11111101010101001000010000101001

Bit	Reset	Description
31	0x1	UTMIP_FSLS_SE1_DRIBBLE_FILTER: One SE1, do not allow dribble
30	0x1	UTMIP_FSLS_SE1_FILTER: Filter SE1
29	0x1	UTMIP_FSLS_SERIAL_SE0_RCV
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

19.13.19.8 USB3_UTMIP_FSLSRX_CFG1_0

UTMIP Full and Low Speed Receive Config 1

Offset: 0x81c | Read/Write: R/W | Reset: 0bxxxxx01000100110011101000000000

Bit	Reset	Description
26	0x0	UTMIP_EARLY_LINE_STATE_FILTER: Assumes line state filtering table is inclusive, not exclusive
25:23	0x4	UTMIP_LS_BOUNCE_LENGTH: Number of clock cycle of LS stable
22:17	0x13	UTMIP_LS_EXTRACTION_COUNT: Phase count on which LS bits are extracted
16:11	0xe	UTMIP_LS_EOP_START_COUNT: Number of SEO clock cycles to block bit extraction
10:5	0x20	UTMIP_LS_SE0_COUNT: Only for this number of 60 MHz of SEO and Idle to end packet
4	0x0	UTMIP_LS_LENIENT_DRIBBLE: Allow for large dribble in low speed mode
3	0x0	UTMIP_FS_LENIENT_DRIBBLE: Allow for large dribble in full speed mode
2	0x0	UTMIP_FS_WEAK_SYNC: Only look for a KK pattern, instead of KJKK
1	0x0	UTMIP_FS_DEBOUNCE: Whether full speed uses debouncing
0	0x0	UTMIP_FS_EOP_LENGTH: Whether full-speed EOP is determined within 3(0) or 4(1) 60 MHz cycles

19.13.19.9 USB3_UTMIP_TX_CFG0_0

UTMIP Transmit Config Signals

Offset: 0x820 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxx00010000001000000000

Bit	Reset	Description
19	0x0	UTMIP_FS_PREAMBLE_J: Output enable sends an initial J before sync pattern
18	0x0	UTMIP_FS_POSTAMBLE_OUTPUT_ENABLE: Output enable turns off 1/2 cycle after
17	0x0	UTMIP_FS_PREAMBLE_OUTPUT_ENABLE: Output enable turns on 1/2 cycle before
16	0x1	UTMIP_FSLS_ALLOW_SOP_TX_STUFF_ERR: Allow SOP to be source of transmit error stuffing
15	0x0	UTMIP_HS_READY_WAIT_FOR_VALID
14:10	0x0	UTMIP_HS_TX_IPG_DLY
9	0x1	UTMIP_HS_DISCON_EOP_ONLY: Only check during EOP
8	0x0	UTMIP_HS_DISCON_DISABLE: Disable high-speed disconnect
7	0x0	UTMIP_HS_POSTAMBLE_OUTPUT_ENABLE: Output enable turns off 1 cycle after
6	0x0	UTMIP_HS_PREAMBLE_OUTPUT_ENABLE: output enable turns on 1 cycle before
5	0x0	UTMIP_SIE_RESUME_ON_LINESTATE: SIE, not macrocell, detects Line State change to resume
4	0x0	UTMIP_SOF_ON_NO_STUFF: SOF when OpMode 3 -- perhaps, when sending controller made packets
3	0x0	UTMIP_SOF_ON_NO_ENCODE: SOF when OpMode 2 -- not likely, for Chirp
2	0x0	UTMIP_NO_STUFFING: No bit stuffing, static programming
1	0x0	UTMIP_NO_ENCODING: No encoding, static programming
0	0x0	UTMIP_NO_SYNC_NO_EOP: Do not send SYNC or EOP

19.13.19.10 USB3_UTMIP_MISC_CFG0_0

UTMIP Miscellaneous Configurations

Offset: 0x824 | Read/Write: R/W | Reset: 0bx00000111110000000000000001111000

Bit	Reset	Description
30:27	0x0	UTMIP_DPDM_OBSERVE_SEL: Select DP/DM obs signals
26	0x0	UTMIP_DPDM_OBSERVE: Use DP/DM as obs bus
25	0x1	UTMIP_KEEP_XCVR_PD_ON_SOFT_DISCON
24	0x1	UTMIP_ALLOW_LS_ON_SOFT_DISCON
23	0x1	UTMIP_FORCE_FS_DISABLE_ON_DEV_CHIRP
22	0x1	UTMIP_SUSPEND_EXIT_ON_EDGE: Suspend exit requires edge or simply a value...
21	0x1	UTMIP_LS_TO_FS_SKIP_4MS: Do not block changes for 4 ms when going from LS to FS (should not happen)
20:19	0x0	UTMIP_INJECT_ERROR_TYPE: Force error insertion into RX path. (Used for IOBIST.) 0 = DISABLE 1 = BIT_ERR 2 = RX_ERR 3 = BIT_RX_ERR
18	0x0	UTMIP_FORCE_HS_CLOCK_ON: Force HS clock always on.
17	0x0	UTMIP_DISABLE_HS_TERM: Force HS termination inactive.
16	0x0	UTMIP_FORCE_HS_TERM: Force HS termination active.
15	0x0	UTMIP_DISABLE_PULLUP_DP: Force DP pull-up inactive. (Overrides FORCE_PULLUP_DP.)
14	0x0	UTMIP_DISABLE_PULLUP_DM: Force DM pull-up inactive. (Overrides FORCE_PULLUP_DM.)
13	0x0	UTMIP_DISABLE_PULLDN_DP: Force DP pull-down inactive. (Overrides FORCE_PULLDN_DP.)
12	0x0	UTMIP_DISABLE_PULLDN_DM: Force DM pull-down inactive. (Overrides FORCE_PULLDN_DM.)
11	0x0	UTMIP_FORCE_PULLUP_DP: Force DP pull-up active.
10	0x0	UTMIP_FORCE_PULLUP_DM: Force DM pull-up active.
9	0x0	UTMIP_FORCE_PULLDN_DP: Force DP pull-down active.
8	0x0	UTMIP_FORCE_PULLDN_DM: Force DM pull-down active.
7:5	0x3	UTMIP_STABLE_COUNT: Number of cycles of crystal clock of signal not changing to consider stable.
4	0x1	UTMIP_STABLE_ALL: Determines if all signal need to be stable to not change a config.
3	0x1	UTMIP_NO_FREE_ON_SUSPEND: Do not use free-running terminations during suspend.
2	0x0	UTMIP_NEVER_FREE_RUNNING_TERMS: Ignore free-running terminations, even when no clock
1	0x0	UTMIP_ALWAYS_FREE_RUNNING_TERMS: Use free-running terminations at all time
0	0x0	UTMIP_COMB_TERMS: Use combinational terminations or synced through CLKXTAL

19.13.19.11 USB3_UTMIP_MISC_CFG1_0

UTMIP Miscellaneous Configurations

Offset: 0x828 | Read/Write: R/W | Reset: 0bx10000000001100110000000000100100

Bit	Reset	Description
30	0x1	UTMIP_PHY_XTAL_CLOCKEN: Selects whether to enable the crystal clock in the module.

Bit	Reset	Description
29	0x0	UTMIP_LINESTATE_BYPASS: Bypass Line State reclocking logic
28	0x0	UTMIP_LINESTATE_NEG: Use negative edge sync for line state
27	0x0	UTMIP_LINESTATE_XCVRSEL3: 0: Use FS filtering on line state when XcvrSel=3 1: Use LS filtering on line state when XcvrSel=3
26:25	0x0	UTMIP_OBS_SEL
24	0x0	UTMIP_FSLS_TDM
23	0x0	UTMIP_FORCE_IOBIST_CLK_ON
22:18	0x6	UTMIP_PLL_ACTIVE_DLY_COUNT: Reserved. Refer to the Clock and Reset registers for this feature.
17:6	0x600	UTMIP_PLLU_STABLE_COUNT: Reserved. Refer to the Clock and Reset registers for this feature.
5	0x1	UTMIP_RX_ERROR_CNT_CLR
4	0x0	UTMIP_RX_ERROR_CNT_EN
3	0x0	UTMIP_FLIP_FSLS_POLARITY
2	0x1	UTMIP_SUSPEND_TERMSEL
1:0	0x0	UTMIP_XCVRSEL3 Bit 0: 0: 0xa5 -> treat as KeepAlive 1: treat as regular packet Bit 1: 0: Turn on FS EOP detection 1: Turn off FS EOP detection

19.13.19.12 USB3_UTMIP_DEBOUNCE_CFG0_0

UTMIP Avalid and Bvalid Debounce

Debounce values IdDig, Avalid, Bvalid, VbusValid, VbusWakeUp, and SessEnd. Each of these signals has its own debouncer, and for each of those, one out of 2 debouncing times can be chosen (BIAS_DEBOUNCE_A or BIAS_DEBOUNCE_B.)

The values of DEBOUNCE_A and DEBOUNCE_B are calculated as follows:

0xffff -> No debouncing at all

ms = *1000 / (1/19.2MHz) / 4

So to program a 1 ms debounce for BIAS_DEBOUNCE_A, BIAS_DEBOUNCE_A[15:0] = 1000 * 19.2 / 4 = 4800 = 0x12c0

Offset: 0x82c | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:16	0xffff	UTMIP_BIAS_DEBOUNCE_B: simulation value -- Used for interrupts
15:0	0xffff	UTMIP_BIAS_DEBOUNCE_A: simulation value -- Used for interrupts

19.13.19.13 USB3_UTMIP_BAT_CHRG_CFG0_0

UTMIP Battery Charger Configuration

Offset: 0x830 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxx000000xx000001

Bit	Reset	Description
13:8	0x0	UTMIP_CHRG_DEBOUNCE_TIMESCALE: Debouncer time scaling, factor-1 to slow down debouncing by. So 0 is 1, 1 is 2, etc.
5	0x0	UTMIP_OP_I_SRC_EN
4	0x0	UTMIP_ON_SRC_EN

Bit	Reset	Description
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

19.13.19.14 USB3_UTMIP_SPARE_CFG0_0

UTMIP Spare Configuration Bits

Offset: 0x834 | Read/Write: R/W | Reset: 0b11111111111111110000000011111000

Bit	Reset	Description
31:0	-65288	UTMIP_SPARE: Spare register bits 0: HS_RX_IPG_ERROR_ENABLE 1: HS_RX_FLUSH_ALAP. Flush as late as possible 2: HS_RX_LATE_SQUELCH. Delay Squelch by 1 CLK480 cycle. 31 to 3: Reserved

19.13.19.15 USB3_UTMIP_XCVR_CFG1_0

UTMIP Transceiver Cell Configuration Register 1

Offset: 0x838 | Read/Write: R/W | Reset: 0bxxxx00000001100010000001000010101

Bit	Reset	Description
27:26	0x0	UTMIP_XCVR_RPU_RANGE_ADJ: 1.5k pull-up resistor range shift
25:24	0x0	UTMIP_XCVR_HS_IREF_CAP: High Speed Iref cap control for bias current stability
23:22	0x0	UTMIP_XCVR_SPARE: Spare bits for USB transceiver pad.
21:18	0x6	UTMIP_XCVR_TERM_RANGE_ADJ: Range adjustment on terminations.
17	0x0	UTMIP_RCTRL_SW_SET: Use a software override on RCTRL instead of automatic bias control
16:12	0x8	UTMIP_RCTRL_SW_VAL: Encoded value to use on RCTRL when software override is enabled, 0 to 16 only
11	0x0	UTMIP_TCTRL_SW_SET: Use a software override on TCTRL instead of automatic bias control
10:6	0x8	UTMIP_TCTRL_SW_VAL: Encoded value to use on TCTRL when software override is enabled, 0 to 16 only
5	0x0	UTMIP_FORCE_PDDR_POWERUP: Force PDDR input into power up.
4	0x1	UTMIP_FORCE_PDDR_POWERDOWN: Force PDDR input into power down. (Overrides FORCE_PDDR_POWERUP.)
3	0x0	UTMIP_FORCE_PDCHRP_POWERUP: Force PDCHRP input into power up.
2	0x1	UTMIP_FORCE_PDCHRP_POWERDOWN: Force PDCHRP input into power down. (Overrides FORCE_PDCHRP_POWERUP.)
1	0x0	UTMIP_FORCE_PDDISC_POWERUP: Force PDDISC input into power up.
0	0x1	UTMIP_FORCE_PDDISC_POWERDOWN: Force PDDISC input into power down. (Overrides FORCE_PDDISC_POWERUP.)

19.13.19.16 USB3_UTMIP_BIAS_CFG1_0

UTMIP Bias Cell Configuration Register 1

Offset: 0x83c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxx00000000101101

Bit	Reset	Description
13:8	0x0	UTMIP_BIAS_DEBOUNCE_TIMESCALE: Debouncer time scaling, factor - 1 to slow down debouncing by. So 0 is 1, 1 is 2, etc.
7:3	0x5	UTMIP_BIAS_PDTRK_COUNT: Control the BIAS cell power down lag. The lag should be 20 µs. For a crystal clock of 13 MHz, it should be set to 5.
2	0x1	UTMIP_VBUS_WAKEUP_POWERDOWN: Reserved. Refer to the PMC registers for this feature.
1	0x0	UTMIP_FORCE_PDTRK_POWERUP: Force PDTRK input into power up.
0	0x1	UTMIP_FORCE_PDTRK_POWERDOWN: Force PDTRK input into power down. (Overrides FORCE_PDTRK_POWERUP.)

19.13.19.17 USB3_UTMIP_BIAS_STS0_0

UTMIP Bias Cell Status Register 0

Offset: 0x840 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	UTMIP_TCTRL: Thermal encoding output from USB bias pad.
15:0	X	UTMIP_RCTRL: Thermal encoding output from USB bias pad.

19.13.19.18 USB3_UTMIP_CHRG_DEB_CFG0_0

UTMIP VDcd_Det and VDat_Det debounce

Debounce values VDcd_Det and VDat_Det. Each of these signals has its own debouncer. For each of those, one out of 2 debouncing times can be chosen (CHRG_DEBOUNCE_PERIOD_A or CHRG_DEBOUNCE_PERIOD_B).

The values of DEBOUNCE_PERIOD_A and DEBOUNCE_PERIOD_B are calculated as follows:

0xffff -> No debouncing at all

ms = *1000 / (1/19.2MHz) / 4

So to program a 1 ms debounce for CHG_DEBOUNCE_PERIOD, we have: CHG_DEBOUNCE_PERIOD[15:0] = 1000 * 19.2 / 4 = 4800 = 0x12c0

Offset: 0x844 | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:16	0xffff	UTMIP_CHRG_DEBOUNCE_PERIOD_B: simulation value -- Used for interrupts
15:0	0xffff	UTMIP_CHRG_DEBOUNCE_PERIOD_A: simulation value -- Used for interrupts

19.13.19.19 USB3_UTMIP_MISC_STS0_0

UTMIP MISC Status Register

Offset: 0x848 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
21	X	UTMIP_FS_DRV_EN: Indicates when the controller is driving on the bus
20:0	X	UTMIP_SPARE_FUSES: Spare Fuses value, to keep the connections preserved

19.13.19.20 USB3_UTMIP_PMC_WAKEUP0_0

UTMIP PMC Wakeup Value

Offset: 0x84c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	R/W	Reset	Description
1	RO	X	UTMIP_LINE_WAKEUP_EVENT_INT_STS: Indicates the status of the PMC wakeup event
0	RW	0x0	UTMIP_LINE_WAKEUP_EVENT_INT_ENB: Interrupt mask for the PMC wakeup event

19.13.19.21 USB3_UHSIC_PLL_CFG0_0

UHSIC PHY PLL Configuration Register 0

Offset: 0xc00 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	Reset	Description
0	0x0	UHSIC_PLL_SPARE: Reserved

19.13.19.22 USB3_UHSIC_PLL_CFG1_0

UHSIC PLL and PLLU Configuration Register 1

This register is used to configure the PHY PLL contained in the UHSIC module as well as the PLLU power up and down.

PLL CONFIGURATION and PARAMETERS

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

Crystal clock -> enters PLLU to generate 12 MHz clock -> enters USB_PHY PLL to generate 480/60 MHz clock

The following parameters control the bring-up of the PLLs:

Coming out of reset or suspend

PllUOnState: start pllu_enable_count and pll_lock_count

Wait ~1 µs to enable the PLL_U (pllu_enable_count == ClkXtal * PLLU_ENABLE_DLY_COUNT * 8)

Wait ~1 ms until PLLU is stable (pll_lock_count == ClkXtal * PLLU_STABLE_COUNT * 256) => USB_PHY PLL_ENABLE

Numbers for a 19.2 MHz crystal clock

- PLLU_ENABLE_DLY_COUNT[4:0] = 1 * 19.2 / 8 = 2.4 = 3 = 0x03
- PLLU_STABLE_COUNT[11:0] = 1000 * 19.2 / 256 = 75 = 0x4b (Note: currently defaults to 0x600)
- XTAL_FREQ_COUNT[11:0] = 2500 * 19.2 / 256 = 187 = 0xbb

Offset: 0xc04 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxx0001100000011000000

Bit	Reset	Description
18:14	0x3	UHSIC_PLLU_ENABLE_DLY_COUNT: 1 µs / (1/19.2MHz) = 19 / 8 = 2.36 = 3

Bit	Reset	Description
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

19.13.19.23 USB3_UHSIC_HSRX_CFG0_0

UHSIC High Speed Receive Config 0

Offset: 0xc08 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxx0010100111000111000

Bit	Reset	Description
18	0x0	UHSIC_NO_STRIPPING: Do not strip incoming data
17:13	0xa	UHSIC_IDLE_WAIT: Number of idle cycles to declare IDLE.
12:8	0xe	UHSIC_ELASTIC_OVERRUN_LIMIT
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.

19.13.19.24 USB3_UHSIC_HSRX_CFG1_0

UHSIC High Speed Receive Config 1

Offset: 0xc0c | Read/Write: R/W | Reset: 0bxxxxxxxx10001000001010010001 0011

Bit	Reset	Description
23:20	0x8	UHSIC_TX_BLOCK_CNT: Controls how long after the end of transmission the receive path is blocked
19:14	0x20	UHSIC_RX_STROBE_DLY_TRIMMER: Number of delays cells between UH_RX_STROBE and RxStrobeClk in zero cycle path
13:9	0x14	UHSIC_INPUT_FIFO_DEPTH: Depth of the 2-bit wide input FIFO. Maximum depth is 20. Can be tuned
8	0x1	UHSIC_LINE_STATE_RESUME_FAKE_SE0: When enabled, send an SE0 for 2 LS symbols at the end of ResumeK
7	0x0	UHSIC_LINE_STATE_BYPASS: Bypass Line State reclocking logic
6	0x0	UHSIC_EARLY_LINE_STATE_FILTER: Assumes line state filtering table is inclusive, not exclusive
5:1	0x9	UHSIC_HS_SYNC_START_DLY: How long to wait before start of sync launches RxActive
0	0x1	UHSIC_HS_ALLOW_KEEP_ALIVE: Allow Keep Alive packets

19.13.19.25 USB3_UHSIC_TX_CFG0_0

UHSIC Transmit Config Signals

Offset: 0xc10 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000

Bit	Reset	Description
9	0x0	UHSIC_HS_READY_WAIT_FOR_VALID

Bit	Reset	Description
8	0x0	UHSIC_PACKET_INVERT_DATA: Invert data during a regular packet
7	0x0	UHSIC_PACKET_FORCE_STROBE_LOW: Force STROBE low during a regular instead of toggling it
6	0x0	UHSIC_HS_POSTAMBLE_OUTPUT_ENABLE: output enable turns off 1 cycle after
5	0x0	UHSIC_SIE_RESUME_ON_LINESTATE: SIE, not macrocell, detects Line State change to resume
4	0x0	UHSIC_SOF_ON_NO_STUFF: SOF when OpMode 3 -- perhaps, when the sending controller made packets
3	0x0	UHSIC_SOF_ON_NO_ENCODE: SOF when OpMode 2 -- not likely, for Chirp
2	0x0	UHSIC_NO_STUFFING: No bit stuffing, static programming
1	0x0	UHSIC_NO_ENCODING: No encoding, static programming
0	0x0	UHSIC_NO_SYNC_NO_EOP: Do not send SYNC or EOP

19.13.19.26 USB3_UHSIC_MISC_CFG0_0

UHSIC Miscellaneous Configurations

Offset: 0xc14 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxx00100010011101000111 10

Bit	Reset	Description
20	0x0	UHSIC_DISABLE_BUSRESET: When 1, the PHY will not send out BusReset during XcvrSelect0, TermSelect0, and Opmode2. It will send out a non-bit-stuffed, non-encoded packet instead.
19	0x0	UHSIC_FORCE_TERMSEL: Value to be forced on TermSelect when FORCE_XCVR_MODE is set.
18	0x1	UHSIC_EXTEND_BK_ACTIVE: Drive the bus keeper one cycle longer when going out of IDLE
17:16	0x0	UHSIC_FORCE_XCVRSEL: Value to be forced on XcvrSelect when FORCE_XCVR_MODE is set.
15	0x0	UHSIC_FORCE_XCVR_MODE: 1: Force the values of XcvrSelect and TermSelect via config bits instead of via the controller
14	0x1	UHSIC_SYMMETRIC_CONNECT_DATA 0: DATA goes high before STROBE goes low and low before STROBE goes high. 1: DATA goes high before STROBE goes low and goes low *after* STROBE goes high.
13	0x0	UHSIC_ASYNC_CONNECT_DATA 0: DATA keeps setup and hold requirements during CONNECT. 1: DATA moves together with STROBE
12	0x0	UHSIC_LONG_CONNECT_STROBE 0: STROBE is 2 periods long during connect. 1: STROBE is 3 periods long during connect
11	0x1	UHSIC_ACTIVE_BK_DRIVE_RX: 1: Use RX state (EOP, etc.) to determine starting time to drive bus keeper instead of waiting for IDLE detection.
10	0x1	UHSIC_ACTIVE_BK_DRIVE_TX: 1: Use TX state to determine starting time to drive bus keeper instead of waiting for IDLE detection.
9	0x1	UHSIC_DETECT_SHORT_IDLE 0: Use 3 edges (negative and positive) to detect an idle state on the line. 1: Use 4 edges.
8	0x0	UHSIC_DETECT_SHORT_CONNECT: 0: Use 3 edges (negative and positive) to detect a connect state on the line. 1: Use 4 edges.

Bit	Reset	Description
7	0x1	UHSIC_SUSPEND_EXIT_ON_EDGE: Suspend exit requires edge or simply a value.
6:5	0x0	UHSIC_INJECT_ERROR_TYPE: Force error insertion into RX path. (Used for IOBIST.) 0 = DISABLE 1 = BIT_ERR 2 = RX_ERR 3 = BIT_RX_ERR
4:2	0x3	UHSIC_STABLE_COUNT: Number of crystal clock cycles of signal not changing to consider stable.
1	0x1	UHSIC_STABLE_ALL: Determines if all signals need to be stable to not change a config.
0	0x0	UHSIC_COMB_TERMS: Use combinational terminations or synced through CLKXTAL

19.13.19.27 USB3_UHSIC_MISC_CFG1_0

UHSIC Miscellaneous Configurations

Offset: 0xc18 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxx100001100000000010

Bit	Reset	Description
17	0x1	UHSIC_PHY_XTAL_CLOCKEN: Selects whether to enable the crystal clock in the module.
16:15	0x0	UHSIC_OBS_SEL: Select which one of 4 observation vectors is presented on the observation bus
14	0x0	UHSIC_FORCE_IOBIST_CLK_ON: Always enable IoBist CLK60. This would be required when you want to use RX_ERROR_CNT_EN.
13:2	0x600	UHSIC_PLLU_STABLE_COUNT: PLLU frequency lock delay.
1	0x1	UHSIC_RX_ERROR_CNT_CLR: Clear IOBST RxError counter
0	0x0	UHSIC_RX_ERROR_CNT_EN: Enable IOBIST RxError counter when not in IOBIST mode. Allows one to read out the number of errors via JTAG during normal operation

19.13.19.28 USB3_UHSIC_PADS_CFG0_0

UHSIC Pads Settings

Offset: 0xc1c | Read/Write: R/W | Reset: 0b0b00000000000000001000100010001000

Bit	Reset	Description
31:24	0x0	UHSIC_HSIC_OPT: Spare config bits
23:20	0x0	UHSIC_TX_SLEWN: Output slew rate (fall time) adjustment
19:16	0x0	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

19.13.19.29 USB3_UHSIC_PADS_CFG1_0

UHSIC Pads Settings

Offset: 0xc20 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxx0011001111101

Bit	Reset	Description
12	0x0	UHSIC_RPU_STROBE: Enable pull up on IO_STROBE
11	0x0	UHSIC_RPU_DATA: Enable pull up on IO_DATA
10	0x1	UHSIC_RPD_STROBE: Enable pull down on IO_STROBE
9	0x1	UHSIC_RPD_DATA: Enable pull down on IO_DATA
8	0x0	UHSIC_LPBK: Internal digital loopback
7	0x0	UHSIC_RX_SEL: 0: Differential read buffers, 1: Single-ended buffers
6	0x1	UHSIC_PD_ZI: Power down single ended receiver
5	0x1	UHSIC_PD_RX: Power down receiver
4	0x1	UHSIC_PD_TRK: Power down tracking circuit
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

19.13.19.30 USB3_UHSIC_CMD_CFG0_0

Determine start-up behavior.

When AUTO_NEGIOTIATE == 0

- Firmware is supposed to take care of things.

HOST

if Opmode1: do not drive anything

else

Initial power up:

- Asynchronously drive 00
- After a few crystal clocks, enable bus keepers

Offset: 0xc24 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000101

Bit	Reset	Description
6	0x0	UHSIC_FORCE_ACTIVATED: Force PHY into activated state without connect handshake (both host and device)
5	0x0	UHSIC_PRETEND_CONNECT_DETECT: While in HOST mode, act as if the input stage has seen a CONNECT pulse from the external PHY
4	0x0	UHSIC_FORCE_RESET: While in HOST mode, force global state machine into RESET state
3	0x0	UHSIC_FORCE_CONNECT: Upon rising value of this bit, force device to send connect. Only useful when AUTO_CONNECT is disabled.
2	0x1	UHSIC_AUTO_CONNECT: As device, automatically send Connect during activation.

Bit	Reset	Description
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.

19.13.19.31 USB3_UHSIC_STAT_CFG0_0

Offset: 0xc28 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	R/W	Reset	Description
31:16	RO	X	UHSIC_CALIOUT
15:8	RO	X	UHSIC_SPARE_STATUS
2:1	RO	X	UHSIC_BUS_STATE
0	RW	0x0	UHSIC_CONNECT_DETECT

19.13.19.32 USB3_UHSIC_SPARE_CFG0_0

UTMIP Spare Configurations, Spare Configuration Bits

Offset: 0xc2c | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:0	-65536	UHSIC_SPARE: Bit 0: HS_RX_IPG_ERROR_ENABLE Bit 1: HS_RX_FLUSH_ALAP Bit 2: FORCE_TRIM_ZERO Bit 3: TX_EN_INIT_CTRL Bits 7 :4: RX_DATA_TRIM[3:0] Bit 11:8: RX_STROBE_TRIM[3:0] Bit 12: FORCE_BK_ON Bit 13: BYPASS_INIT_BLOCK Bit 14: FORCE_SM_IDLE Bit 15: FORCE_BK_OFF Bit 16: ALLOW_FLUSH_SHIFT

19.13.19.33 USB3_UHSIC_MISC_STS0_0

UHSIC SPARE Fuse Value

Offset: 0xc30 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
0	X	UHSIC_TX_HS_VLD: Indicates when the controller is driving on the bus

19.13.19.34 USB3_UHSIC_PMC_WAKEUP0_0

UHSIC PMC Wakeup Value

Offset: 0xc34 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	R/W	Reset	Description
1	RO	X	UHSIC_LINE_WAKEUP_EVENT_INT_STS: Indicates the status of the PMC wakeup event
0	RW	0x0	UHSIC_LINE_WAKEUP_EVENT_INT_ENB: Interrupt mask for the PMC wakeup event

19.13.19.35 USB2_QH_USB2D_QH_EP_n_OUT_0

USB2D Queue Head for OUT Endpoint n

There are 16 USB2D Queue Head for OUT Endpoint registers, where n = 0 through 15.

Offset: 0x1000 + (n * 0x80) | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint n. 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 n.

19.13.19.36 USB2_QH_USB2D_QH_EP_n_IN_0

USB2D Queue Head for IN Endpoint 0

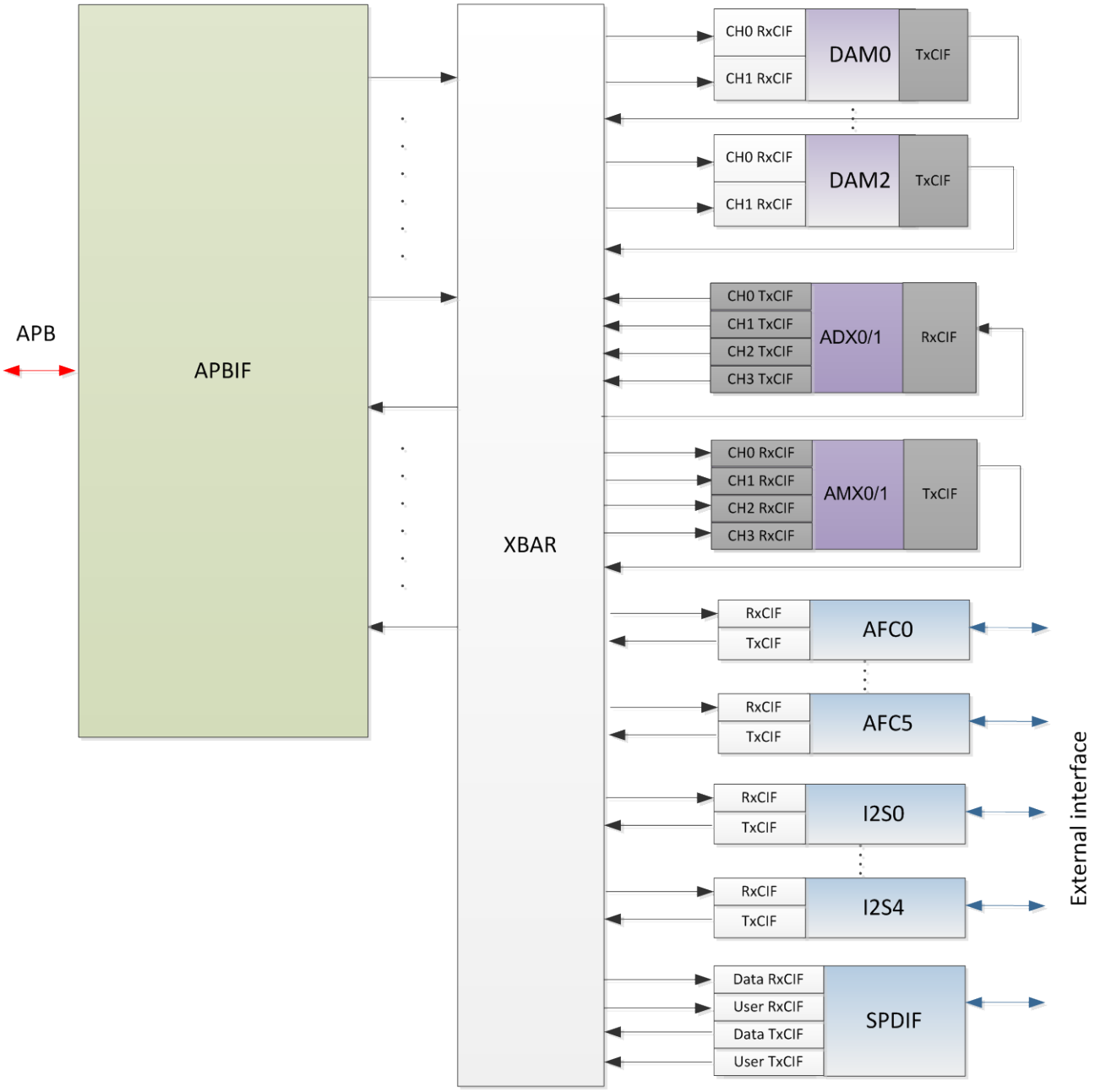
Offset: 0x1040 + (n * 0x80) | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint n. 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 n.

20.0 AUDIO HUB (AHUB)

The audio hub (AHUB) in Tegra® K1 mobile processors has external I2S and S/PDIF interfaces. It can process audio for Digital Audio Mixers (DAMs), Audio Multiplexers (AMXs), Audio Demultiplexers (ADXs), Audio Flow Controllers (AFCs), and DMA (APBIF-DMA) channels to communicate with the memory. The AHUB supports multiple interfaces to the audio devices in the system cellular baseband; different audio codecs; Bluetooth modules; A/V receivers; etc. The AHUB can support the different interface, protocol, and signal quality requirements of these audio devices.

Figure 43: Audio Hub Functional Block Diagram



The Audio Hub (AHUB) consists of

- 10 APBIF DMA channels
- 5 I^2S controllers
- 1 S/PDIF controller
- 3 Digital Audio sample-rate conversion and Mixing blocks (DAMs)
- 2 Audio Multiplexers (AMX) and 2 Audio Demultiplexers (ADXs) for audio processing.
- 6 Audio Flow Controllers (AFCs)

20.1 Crossbar

The crossbar (XBAR) is a 36x39 full crossbar switch with unidirectional data flow, with 36 TX and 39 RX clients. The XBAR has multicasting capability: 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. It also has the ability to connect TX clients with Rx clients in many sessions that function independently of each other.

20.1.1 XBAR Programming Guidelines

To connect two modules in AHUB, for example, DAM0's output to I2S1's input:

1. Go to the register corresponding to the receiver, in this case, the I2S1 AUDIO_I2S1_RX0_0 register.
2. Enable the bit that corresponds to the transmitter, in this case, the field DAM1_TX0.
3. Ensure that all other fields in the AUDIO_I2S1_RX0_0 register are disabled. At power on, all the fields are disabled. So this step is required only while setting up the audio routing after removing the previous use case's audio routing.

20.2 Audio Client Interface (ACIF)

Audio streams are routed through the AHUB by interconnecting various modules using the Audio Client Interface (ACIF).

All the AHUB modules have one or more Transmit Audio Client Interfaces (TxCIFs) and one or more Receive Audio Client Interfaces (RxCIFs). Hence all modules in the AHUB have registers named <module>_AUDIOCIF_<tag>_CTRL that are used to configure the interfaces. The audio CIF interface includes common functionalities that are used in all the AHUB modules such as:

- Converting the number of channels in the input stream(s) before the stream is fed into the module. This feature is limited to converting a mono stream to a stereo stream and vice-versa.
- Converting the number of bits/sample for the input stream(s) before the stream is fed into the module.

20.2.1 General ACIF Register Specification

The ACIF registers are present in the AHUB modules. For convenience of reference for the programming guidelines below, here is a sample ACIF register.

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10

Bit	R/W	Reset	Description
			10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV:

Bit	R/W	Reset	Description
			0 = ZERO 1 = COPY

20.2.2 ACIF Programming Guidelines

The following programming guidelines are common for the ACIF registers in all AHUB modules. The programming guidelines of the respective modules will refer to this subsection when it comes to programming the ACIF registers.

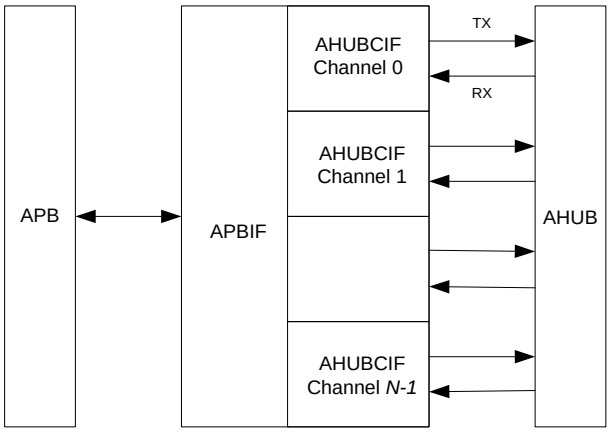
1. Set the AUDIO_CHANNELS field to the number of channels in the input or output stream of a module, depending on whether the ACIF in question is an RxCIF or TxCIF respectively
2. Set the CLIENT_CHANNELS field to the number of channels of the stream as dealt with inside the module. This setting is different from AUDIO_CHANNELS only when a conversion of the number of channels of an audio stream is desired.
3. Similarly, set the AUDIO_BITS and CLIENT_BITS fields.
4. If AUDIO_BITS and CLIENT_BITS do not match, the EXPAND and TRUNCATE fields are used to determine how to transition from AUDIO_BITS to CLIENT_BITS (in the case of a RxCIF) or from CLIENT_BITS to AUDIO_BITS (in the case of an TxCIF). Expanding always results in the actual data being shifted to the MSB bits and the rest filled with zeros, ones, or some random bits from an LFSR. TRUNCATE can be set to chopping or rounding.
5. If AUDIO_CHANNELS and the CLIENT_CHANNELS are different, the STEREO_CONV field is used to determine how a stereo stream is converted to a mono stream, and the MONO_CONV field is used to determine how a mono stream is converted to a stereo stream.

20.3 APBIF

Note: Tegra K1 devices contain identical APBIF and APBIF2 functional blocks. Unless specified otherwise, references to APBIF in this document also apply to APBIF2. Each block has its own register set.

The AMBA Peripheral Bus Interface (APBIF) is the agent for the APB control flow and DMA operation, which sends or receives data from/to Memory. The APBIF is composed of *N* instances of AHUBIF and APB interfaces as shown in the figure below.

Figure 44: APBIF Block Diagram



20.3.1 APBIF 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

20.3.2 APBIF Programming Guidelines

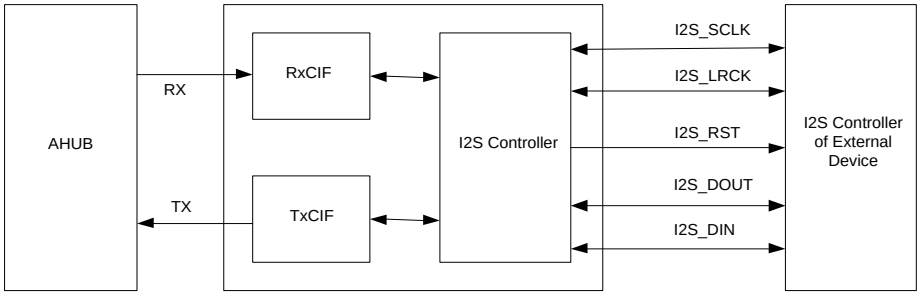
Note: APBIF is comprised of APBIF1 (channels 0 to 3) and APBIF2 (channels 4 through 9). Refer to the APBIF and APBIF2 register descriptions for more information.

1. Program the CLK_RST_CONTROLLER_CLK_OUT_ENB_V register to enable Audio XBAR and APBIF
2. Program the CLK_RST_CONTROLLER_CLK_SOURCE_AUDIO_0 register. Set the AUDIO_CLK_SRC field to CLK_SRC_ALT if an alternate clock source needs to be chosen followed by setting of AUDIO_CLK_SRC_RATE to choose the alternate clock source for DAM.
3. Deassert APBIF and AUDIO reset by programming the CLK_RST_CONTROLLER_RST_DEVICES_V register.
4. Wait for resets by polling until the reset bit clears.
5. Configure the ACIF interfaces of the APBIF according to the programming guidelines available in the ACIF Programming Guidelines section.
6. Configure the control register for the APBIF Tx and Rx channels. APBIF_CHANNEL0_CTRL_0 is the control register for both Tx and Rx channel 0 (similar control registers are present for all 10 APBIF channels). Configuration options for the Tx channels are :
   - TX_ENABLE: enables the Tx channel
   - TX_THRESHOLD: sets the FIFO threshold for sending the tx_req signal to APBDMA
   - TX_PACK_EN: enables Tx packed mode
   - TX_PACK: pack options of four 8-bit words or two 16-bit words. The options are similar for Rx channels as well
   - LOOPBACK: enables APBIF internal loopback (debugging feature).

20.4 I²S Controller

The Inter-IC Sound (I^2S) controller implements full-duplex and bidirectional and single direction point-to-point serial interfaces. It can interface with I^2S-compatible products, such as compact disc players, digital audio tape devices, digital sound processors, modems, Bluetooth chips, etc.

Figure 45: I2S Interaction with External Device



The I^2S controller can operate both as master and slave. It supports the following data transfer modes:

- I^2S mode
- Left Justified Mode (LJM)
- Right Justified Mode (RJM)

- DSP mode, as defined in the Philips inter-IC-sound (I^2S) bus specification
- PCM mode with short (one-bit-clock wide) and long-fsync (two bit-clocks wide
- Network (Telephony) mode with independent slot selection for both Tx and Rx
- TDM mode with flexibility in number of slots with up to 16 slots.
- Capability to drive-out a High-Z outside the prescribed slot for transmission

The I^2S controller can transmit and receive word lengths of 8, 16, 24, and 32.

It supports u-Law and A-Law compression/decompression.

The I^2S controller can control the flow of traffic from another I^2S controller operating on an independent bit clock (with a ppm difference compared to its own bit clock).

20.4.1 Transmission/Reception Data Formats

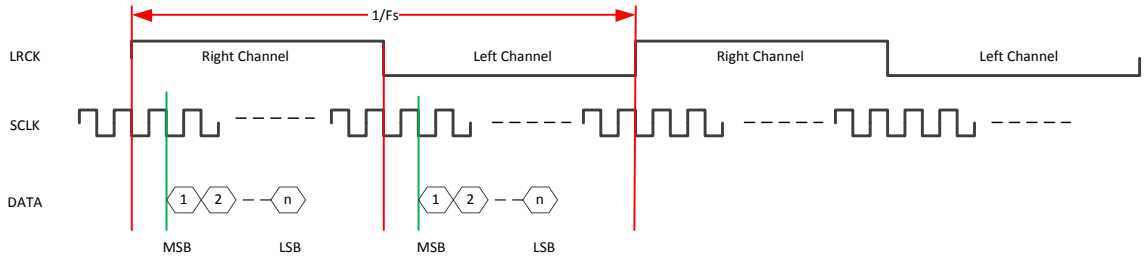
20.4.1.1 LRCK Modes

This subsection illustrates three LRCK modes: Basic I2S mode, Right Justified mode, and Left Justified mode.

Basic I2S Mode

In Basic I2S mode, data starts one sclk after the LRCK edge (offset = 1).

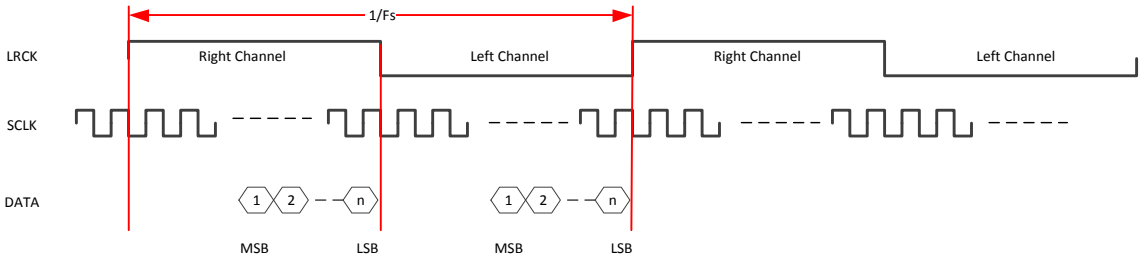
Figure 46: Basic I2S Mode



Right Justified (RJ) Mode

In RJ mode, data starts (r/2 – n) SCLKs after the LRCK edge (offset = r/2 – n, where r = number of SCLKs per LRCK, n = number of bits/sample).

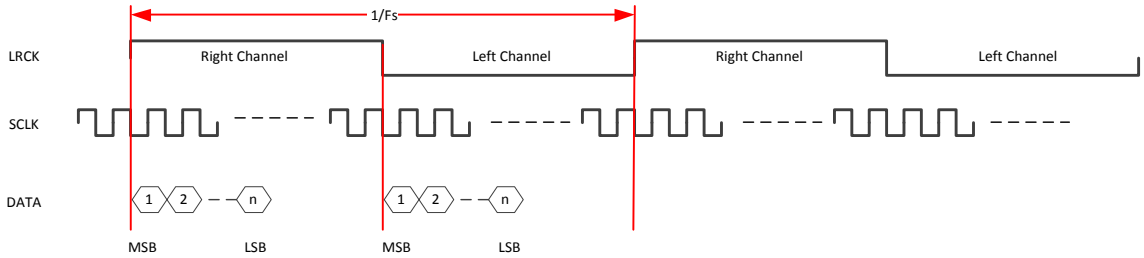
Figure 47: RJ Mode



Left Justified (LJ) Mode

In LJ mode, data starts 0 SCLKs after the LRCK edge (offset = 0).

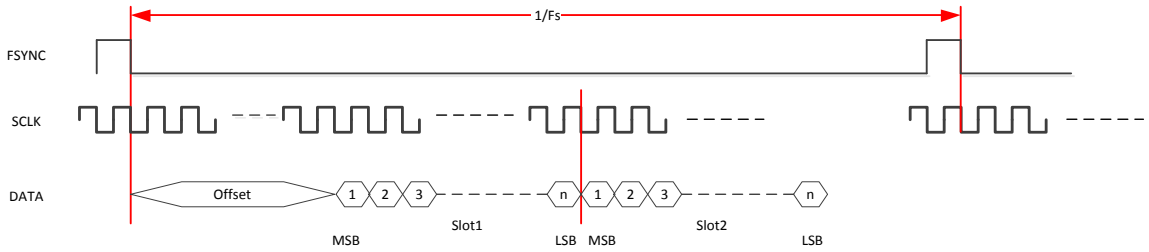
Figure 48: LJ Mode



FSYNC Modes

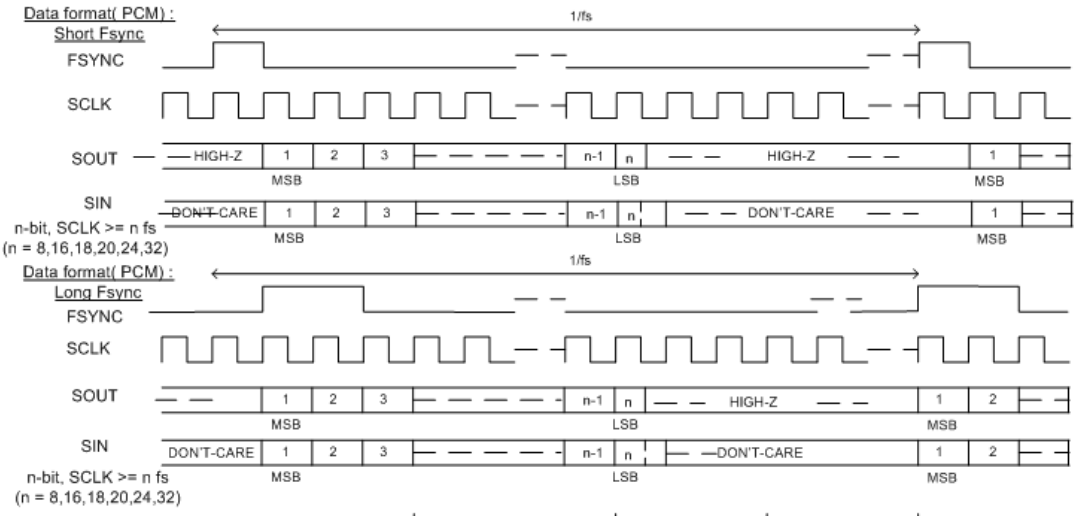
The width of the FSYNC, the offset value, number of slots and number of SCLKs per 1/Fs are all configurable. The slots are always contiguous.

Figure 49: FSync Mode

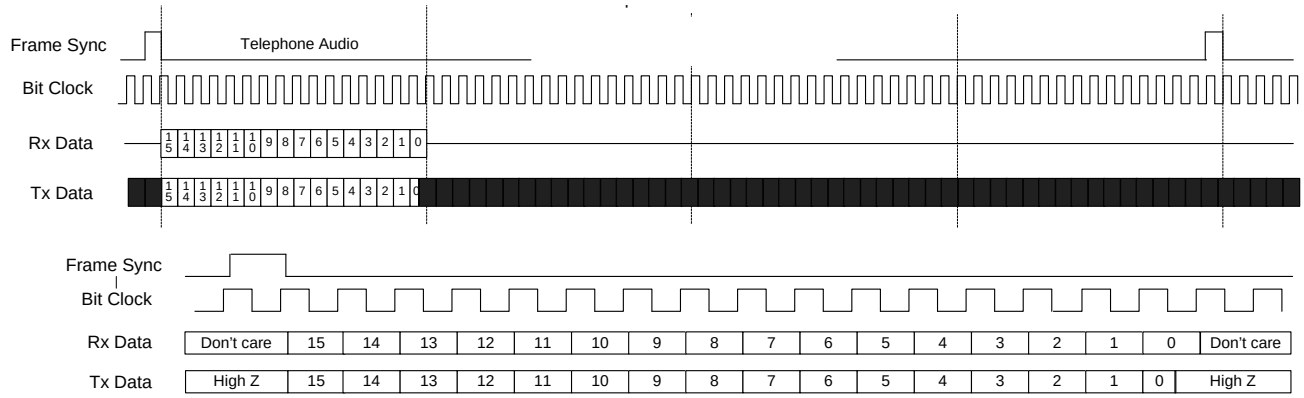


PCM Mode

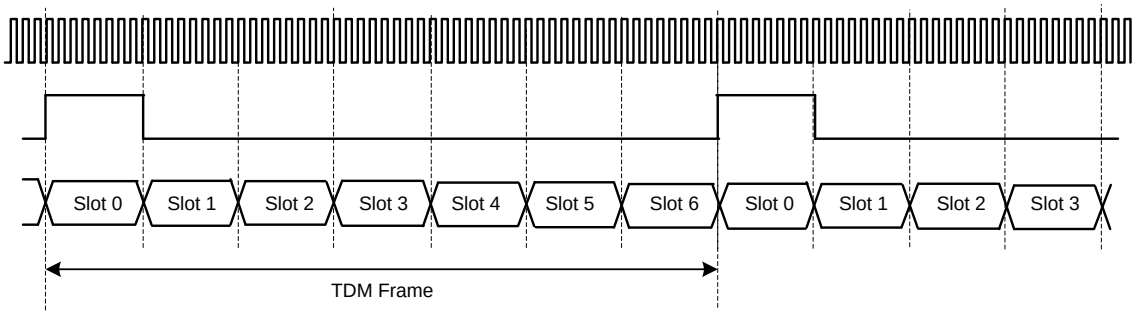
Figure 50: PCM Mode



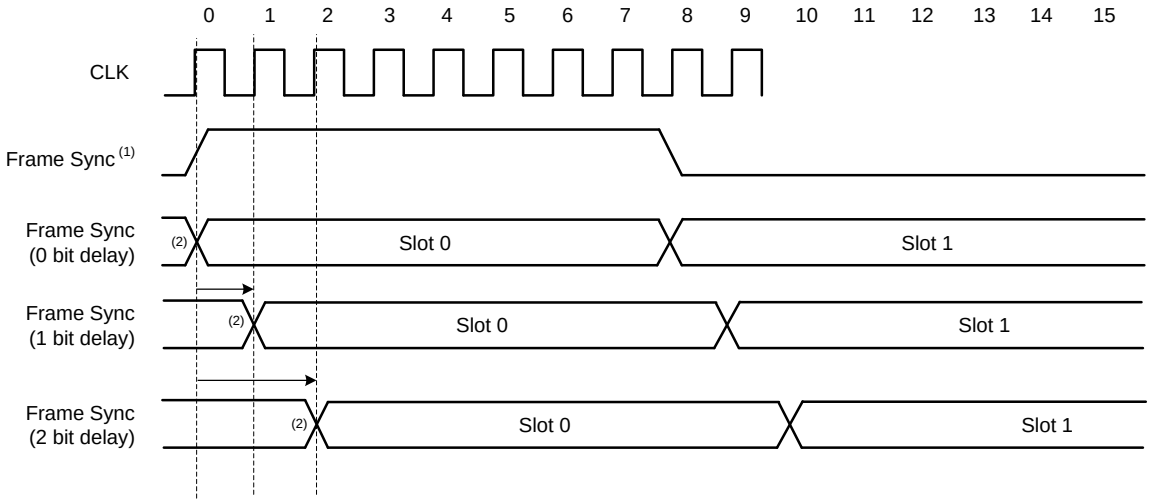
Data Formats in NW Mode



Data Format in TDM Mode



1. FS duration of a slot is shown. FS duration of a single bit is also supported.



1. FS duration of a slot is shown. FS duration of a single bit is also supported.
2. Last bit of the last slot of the previous frame. No gap is allowed between this bit and the first bit of slot 0.

20.4.2 I2S Controller Programming Guidelines

20.4.2.1 Module Initialization

1. Program the CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0 register to enable the I2S clock.
2. Program the CLK_RST_CONTROLLER_CLK_SOURCE_I2S0_0 register (for I2S0) and similar registers for other instances to program the I2S clock source. Set I2S0_CLK_SRC to select the clock source.
3. Choose SYNC_CLK as the clock source in the I2S0_CLK_SRC field of the CLK_RST_CONTROLLER_CLK_SOURCE_I2S0_0 register to select alt clock sources.
4. This should be followed by setting the CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S0_0 register to choose the alternate clock source for I^2S.
5. Deassert I2S0 and AUDIO resets by programming the CLK_RST_CONTROLLER_RST_DEVICES_L register.
6. Wait for resets to happen before proceeding further.
7. Configure the ACIF registers I2S_AUDIOCIF_I2SRX_CTRL_0 and I2S_AUDIOCIF_I2STX_CTRL_0 by following the guidelines in the ACIF Programming Guidelines subsection.
8. Enable Rx transmit or Tx transmit by configuring I2S_CTRL_0.

20.4.2.2 I2S Sampling Rate Selection

The channel_bit_cnt can be calculated using the equation:

channel_bit_cnt = (frequency of bit_clk) / (2 * required sampling rate) -1

If this calculation returns a fractional value, use the non-symmetry feature of the controller to attain the required sampling rate. In that case, the channel_bit_cnt value should be programmed with an integer that is closest to the fraction, but less than the fraction.

The table below contains some examples for common sampling rates with CLK_SOURCE_I2S = 24 MHz:

Sampling Rate	**I2S_NEW_TIMING[12:00] (mark-space ratio:: Left_channel : Right_channel)**				
	CLK_DIVISOR=0 BIT_CLK=24 MHz	CLK_DIVISOR=1 BIT_CLK=12 MHz	CLK_DIVISOR=3 BIT_CLK=6 MHz	CLK_DIVISOR=5 BIT_CLK=4 MHz	CLK_DIVISOR=7 BIT_CLK=3 MHz
8 KHz	0x05DB	0x02ED	0x0176	0x00F9	0x10BB
	(1500:1500)	(750:750)	(375:375)	(250:250)	(187:188)
32 KHz	0x0176	0x10BB	Not supported	0x103E	Not supported
	(375:375)	(187:188)		(62:63)	
44.1 KHz	0x010F	0x0087	0x0043	0x002C	0x0021
	(272:272)	(136:136)	(68:68)	(45:45)	(34:34)
48 KHz	0x00F9	0x007C	0x103E	Not supported	Not supported
	(250:250)	(125:125)	(62:63)		
96 KHz	0x007C	0x103E	0x101F	Not supported	Not supported
	(125:125)	(62:63)	(31:32)		

Notes:

- Ideally, any sampling rate can be generated, either accurately or approximately with any clk_src selected for I2S, if the clk_divisor and the channel_bit_cnt are programmed accordingly.
- The NON_SYM.EN feature is meant to create the sampling rates approximately by realizing an odd bit rate with a non-50:50 mark/space ratio. However, if the bit rate (2*channel_bit_cnt) itself is a fraction, the resultant sampling rate will not be accurate. The entries shown as not supported in the above table are attributed to such a deviation.
- When the channel_bit_cnt is less than 32, the bit_size should be programmed to be less than channel_bit_cnt.

20.4.2.3 Programming I2S to Operate in Various Modes

After the module initialization (reset and clock programming), program the corresponding bits of the following registers.

Table 71: I2S Programming for Various Modes

Register	Basic	LJM	RJM	DSP	PCM	NW	TDM
FRAME_FORMAT	0	0	0	1	1	1	1
LRCK_POLARITY	0	1	1	1	1	1	1
CHANNEL_BIT_CNT	Number of bit clocks in left channel	Number of bit clocks in left channel	Number of bit clocks in left channel	Number of bits in left channel + right channel	Number of bit clocks per frame	Number of bit clocks per frame	Number of bit clocks per frame
TX_DATA_OFFSET	1	0	CHANNEL_BIT_CNT -BITS_PER_SAMPLE	1	1: Short Fsync 0: Long Fsync	1	0/1/2
RX_DATA_OFFSET	1	0	CHANNEL_BIT_CNT – BITS_PER_SAMPLE	1	1:short Fsync 0:Long Fsync	1	0/1/2
FSYNC_WIDTH	Number of bit clocks in left channel	Number of bit clocks in left channel	Number of bit clocks in left channel	1	0: Short Fsync 1: Long Fsync	0	Bits per sample (slot size)
HIGHZ_CTRL	0	0	0	0	2	2	0/1/2
EDGE_CTRL	0	0	0	0	1	1	1
TOTAL_SLOTS	0	0	0	1	0	3	0,1..7
SLOT_ENABLES	0x01	0x01	0x01	0x03	0x01	*	**

* In NW mode, there can only be one active slot. So 0x01, 0x02, 0x04, 0x08 are allowed values.

** In TDM mode, there is no restriction on what/how many slots are enabled.

The only restriction on the programming sequence is that all other registers must be programmed first before the Tx/Rx channels are enabled (which is done by setting XFER_EN_TX/XFER_EN_RX).

20.4.2.4 Enabling I2S Interrupts

1. Enable the MONITOR_INT_EN bit in the I2S_FLOW_STATUS_0 register.
2. Enable the I2SX_TXCIF_OVERRUN and I2SX_RXCIF_UNDERRUN bits in the APBIF_I2S_INT_MASK_0 register.

20.4.2.5 General Guidelines

- Always enable I2S interrupts and monitor overruns in TxCIF and underruns in RxCIF. When any one of them occurs, apply soft reset to I2S before using it again.
- Enable flow control only when the signals originate (at least partly) from one I2S and go through another I2S. Use the AFC controller to do this.

20.5 Serial Peripheral Device Interface (S/PDIF)

This interface is primarily intended to carry audio data coded other than as linear PCM coded audio samples. Provisions are also made to allow the interface to carry data related to computer software or signals coded using non-linear PCM. The format specification for these applications is not part of this standard.

20.5.1 S/PDIF Features

- Fully implements the IEC-958 standard interface
- Provides mono and stereo support for sampling frequencies 32 kHz, 44.1 kHz, and 48 kHz
- Supports the following data formats.
  - 16-bit
  - 20-bit
  - 24-bit
  - Raw
- Supports “autolock” mode to automatically detect the “spdifin” sample rate and lock onto the data stream.
- Supports “override” mode to provide a manual control to sample the “spdifin” data stream.
- Provides a loopback mode to route the “spdifout” back to “spdifin” for self-testing.

20.5.2 S/PDIF Programming Guidelines

20.5.2.1 16-, 20-, 24-bit Modes

During transmission, the audio data is taken from the TX data FIFO, the channel status bit is taken from the TX channel status page buffer, and the user status bit is from the TX user FIFO. The preamble bits are generated by the hardware. The Valid bit is always zero.

During reception, the audio data is stored in the RX data FIFO, the channel status bit is stored in the RX channel status page buffer and the user status bit is stored in the RX user FIFO.

In addition to storing the audio data, the RX data FIFO also stores the preamble bits, channel status bit, user status bit and the valid bit. The reception of the channel bits start after the B-preamble is detected.

The audio-data sample has 16, 20, or 24 bits of data. Firmware has to transmit 16-, 20-, or 24-bit data and read 16-, 20-, or 24-bit data.

20.5.2.2 Raw Mode

During transmission, the audio data, the preamble bits, channel status bits, the user bits, and the valid bits are transmitted from the RX data FIFO. The firmware has to provide the correct preamble bits for the receiving device to synchronize with the transmitter. The firmware has to provide preambles according to the following table:

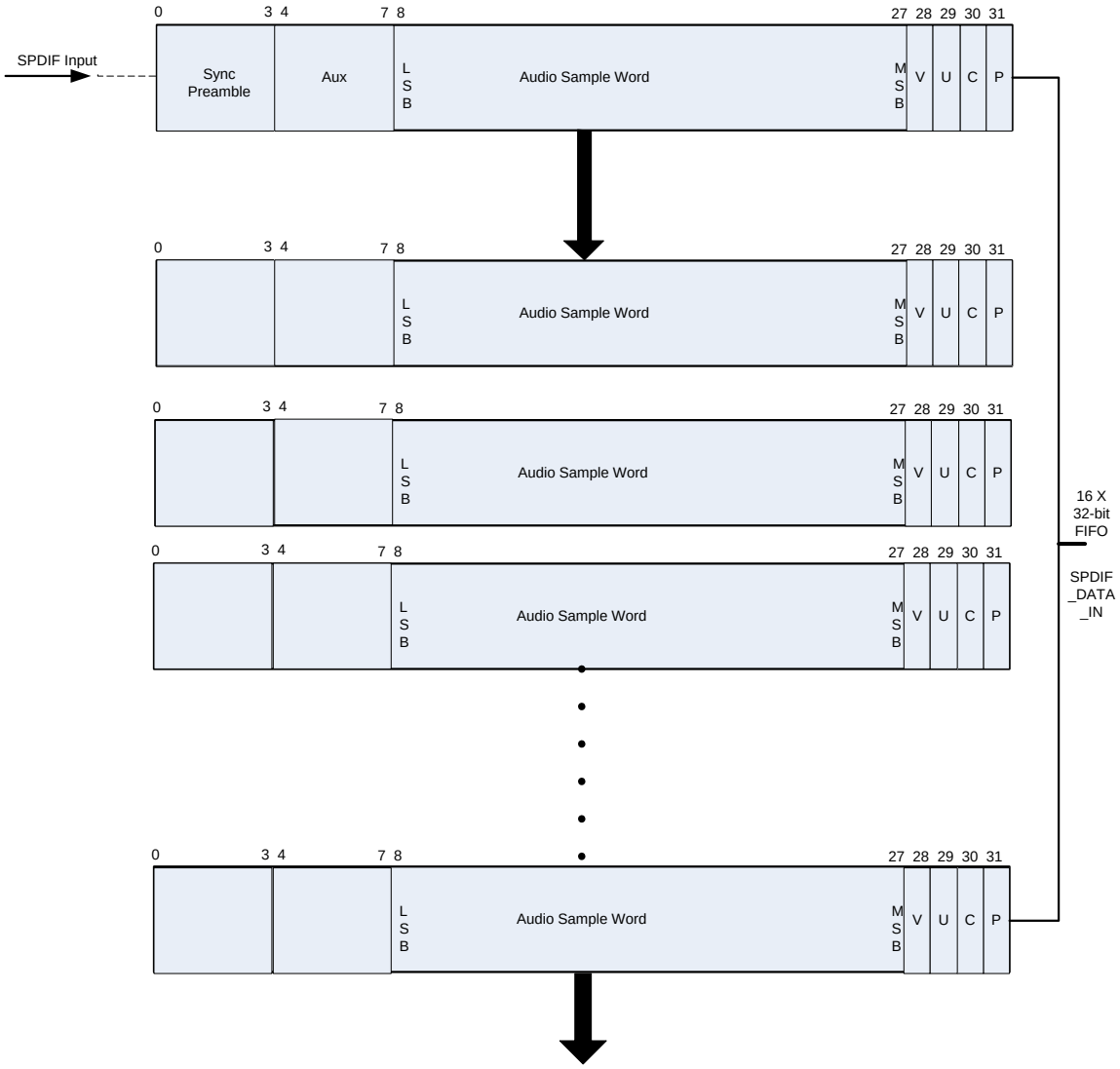
Preceding State in the Parity Bit: 0/1

Table 72: 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 51: S/PDIF Data Format for Raw Mode



20.5.2.3 Module Reset

The application may initialize or reset the S/PDIF module via the clock and reset (CAR) control's RST_DEVICES_L_0 register.

20.5.2.4 Clock Source/Divider Control

The SPDIFOUT clock source/divider can be selected by programming CAR's CLK_SOURCE_SPDIF register. Refer to the S/PDIF registers below for the needed input frequency for the desire output audio sample rate.

The SPDIFIN clock source/divider can be selected by programming CAR's CLK_SOURCE_SPDIF register. Refer to the SPDIF registers below for the needed oversample frequency.

20.5.2.5 Clock Enable Control

The application may enable the S/PDIF clocks by programming the CAR's CLK_OUT_ENB_L_0 register.

20.5.2.6 Detecting Incoming Sampling Rate

The SPDIFIN is capable of detecting and locking onto the ‘spdifin’ data stream regardless of the sample rate. It achieves this by using an oversampling technique.

In addition to reading the sampling rate from the channel status information, one can also read the PERIOD field in the STROBE_CTRL_0 register to estimate the incoming sample rate. The last two digits of PERIOD indicate the fractional clock period.

Sample rate ≈ (‘spdifin’ clock frequency * 1000) / (PERIOD * 128)

If using the PERIOD method, it is better to read this field toward the end of the song or just before turning off the SPDIFIN. Although the hardware can lock onto the SPDIFIN data stream in as little as 2 subframes of time, the hardware is actually constantly re-adjusting itself (due to jitter, synchronization loss, etc. in the data stream) to provide the strobe point to the center of the biphase data stream. Over time, the strobe point will vary less and less because the hardware has already adjusted to all the variations seen in the data stream.

20.5.2.7 S/PDIF Transmit Data Flow

Make sure that the TX_FIFO is filled before enabling the transmission by setting the TX_EN field of CTRL_0 register. Never allow the TX_FIFO to go empty at any time during transmission. The interrupt is generated when the Tx Data FIFO empty count is greater than the Tx DATA attention value (TX_ATN_LVL field of DATA_FIFO_CSR_0 register) if the Tx interrupt enable is set.

20.5.2.8 S/PDIF Receive Data Flow

To enable the reception of RX_FIFO, set the RX_EN field of the CTRL_0 register. Never allow the RX_FIFO to become full at any time during reception. The interrupt is generated when the Rx Data FIFO data count is greater than the Rx DATA attention value (RX_ATN_LVL field of the DATA_FIFO_CSR register), if the Rx interrupt enable bit is set.

20.5.2.9 Method of Filling/Retrieve Data from TX/RX User/Data FIFO

Use the DMA method by programming the APB DMA and APBIF registers. With this method, the FIFO attention levels need to be set in the APBIF just like in method 2. But instead of enabling interrupts when the FIFO attention level is reached, one should program the APB DMA registers. Now, every time the attention level is reached, a DMA request will be generated by the APBIF to the APB DMA and the APB DMA will perform the data moving task from/to APBIF FIFOs to the AHB bus.

20.6 Digital Audio Mixers

The digital audio mixers (DAMs) can mix up to two input stereo streams of different frame rates and sample sizes and produce one output audio stream.

The Sample Rate Converter converts the frame rate of one of the input channels using a collection of Low-Latency Filters (LLFs):

- Input/output frequencies supported: 8, 11.025, 16, 22.05, 44.1, 48, 88.2, 96, 176.4, and192kHz
- Mono/Stereo audio for both input channels is supported. Stereo SRC is supported for channel 0. Channel 1, which is the bypass channel, does not support SRC.
- Up-conversion and down-conversion from any supported frequency to any supported frequency
- 8-, 16-, 24-, 32-bit audio data
- Fixed latency on round trip between baseband and codec of ~7 ms.

The bit-width converter at the CIF should convert the input stream to 32 bits because SRC operates only on 32-bit data

A programmable gain controller is available in both input channels. It can be programmed to give a linear gain between -8 and 7.9998.

The DAM Mixer mixes the output from channel 0 (stereo SRC channel) with channel 1 (bypass channel). The Mixer handles saturation and has the following mixing options:

- Both channels wait on each other
- Channels wait on none
- Channel 0 waits on channel 1
- Channel 1 waits on channel 0

20.6.1 DAM Programming Guidelines

1. Program the CLK_RST_CONTROLLER_CLK_OUT_ENB_V register to enable the DAM clock.
2. Program CLK_RST_CONTROLLER_CLK_SOURCE_DAM0_0 (for DAM0) and similar registers for other instances to program the DAM clock source. Set the DAM0_CLK_SRC field to CLK_SRC_ALT, if an alternate clock source is needed followed by setting DAM0_CLK_SRC_RATE to choose the DAM alternate clock source.
3. Deassert DAM0 and AUDIO reset by programming the CLK_RST_CONTROLLER_RST_DEVICES_V register.
4. Poll for this bit to clear.
5. Configure the CIF registers (DAM_AUDIOCIF_CH0_CTRL_0, DAM_AUDIOCIF_CH1_CTRL_0, and DAM_AUDIOCIF_OUT_CTRL_0) according to the guideline given in the ACIF Programming Guidelines subsection.
6. Set the input FSIN by programming DAM_CH0_CTRL for channel 0 and DAM_CH1_CTRL for channel1.
7. Set the Gain for both the channels by setting DAM_CH0_CONV and DAM_CH1_CONV for both channel 0 and channel 1, respectively.
8. Enable the DAM output by setting FSOUT into the DAM_CTRL register.
9. If the use case needs a farrow filter in its pipeline, enable the farrow filter by configuring the DAM_FARROW_PARAM register.
10. Software can manually program the coefficient RAM by setting the COEF_RAM_EN bit to 0 in the DAM_CH0_CTRL register. The values for each of the 8 use cases are stored statically in hardware and can be accessed without any software programming by setting COEF_RAM_EN bit to 1 in DAM_CH0_CTRL register.
    - For programming the coefficient RAM with the HW_ADR_EN bit unset in the DAM_AUDIORAMCTL_DAM_CTRL register, set the coefficient RAM offset in the RAM_ADR field of the DAM_AUDIORAMCTL_DAM_CTRL register and set data in the DATA field in DAM_AUDIORAMCTL_DAM_DATA.
    - For programming the coefficient RAM with the HW_ADR_EN bit set in the DAM_AUDIORAMCTL_DAM_CTRL register, set the coefficient RAM offset in the RAM_ADR field of the DAM_AUDIORAMCTL_DAM_CTRL register to 0 and set data in DATA field in DAM_AUDIORAMCTL_DAM_DATA. Hardware auto-increments RAM_ADR after each write.
11. For channel mixing, the DATA_SYNC field in both DAM_CH0_CTRL and DAM_CH1_CTRL have to be configured correctly to have mixing in 4 available modes:
    - Ch0 wait on Ch1
    - Ch1 wait on Ch0
    - Both channels wait on each other (preferred option)
    - Channels do not wait
12. Set STEREO_SRC_EN in the DAM_CTRL register to enable stereo SRC.
13. Enable the DAM by setting the ENABLE bit in DAM_CTRL.

For details on the fields within each DAM register, refer to the “DAM Registers” subsection.

20.7 Audio Multiplexer Block (AMX)

The Audio Multiplexer Block (AMX) can multiplex up to 4 input streams of up to 16 channels each and generate an output stream of up to 16 channels

A “byte RAM” helps to form an output frame by any combination of bytes from the 4 input frames

Two modes for data synchronization between input frames are available:

- Wait For All mode: At the beginning, wait for all enabled input streams to have data before forming the very first frame.
- Wait for Any mode: Start whenever data is available in any one of the enabled input streams.

In either mode, once the first output frame is sent out, AMX always waits for all active streams to have data available before forming and sending subsequent frames.

20.7.1 AMX Programming Guidelines

1. Configure Audio CIF control registers AMX_AUDIOCIF_CH0_CTRL, AMX_AUDIOCIF_CH1_CTRL, AMX_AUDIOCIF_CH2_CTRL, AMX_AUDIOCIF_CH3_CTRL, and AMX_AUDIOCIF_OUT_CTRL according to the guidelines given in the ACIF Programming Guidelines subsection.
2. Configure the CH_DEP parameter field in the AMX_CTRL_0 register to determine whether to wait for all enabled channels to have data before transfer or start sending the data when any one of the input channels has data (WT_ON_ALL & WT_ON_ANY).
3. Configure the MSTR_CH_NUM register for designated master channel, so that if the data is available on a particular channel, the output can be sent.
4. Enable the AMX output by setting BYTE_EN in AMX_OUT_BYTE_EN0 and AMX_OUT_BYTE_EN1.
5. For Byte RAMCTL programming:
   - For programming RAM with the HW_ADR_EN bit cleared in the AMX_AUDIORAMCTL_AMX_CTRL register, set the RAM offset in the RAM_ADR field of the AMX_AUDIORAMCTL_AMX_CTRL register and set data in the DATA field in the AMX_AUDIORAMCTL_AMX_DATA register.
   - For programming RAM with the HW_ADR_EN bit set in the AMX_AUDIORAMCTL_AMX_CTRL register, set the RAM offset in the RAM_ADR field of the AMX_AUDIORAMCTL_AMX_CTRL register to 0 and set data in the DATA field in the AMX_AUDIORAMCTL_AMX_DATA register. Hardware auto-increments the RAM_ADR after each write.

20.8 Demultiplexer Block (ADX)

The demultiplexer block (ADX) takes an input stream with up to 16 channels and demultiplexes it into four output streams of up to 16 channels each.

A “byte RAM” helps to form output frames by any combination of bytes from the input frame. Its design is identical to that of the byte RAM in the AMX except that the data flow direction is reversed.

20.8.1 ADX Programming Guidelines

1. Configure the Audio CIF control registers ADX_AUDIOCIF_IN_CTRL, ADX_AUDIOCIF_CH0_CTRL, ADX_AUDIOCIF_CH1_CTRL, ADX_AUDIOCIF_CH2_CTRL, and ADX_AUDIOCIF_CH3_CTRL according to the guidelines given in the ACIF Programming Guidelines subsection.
2. Configure ADX_OUT_CH_CTRL_0 for enabling or disabling four output channels.
3. Enable the ADX input by setting BYTE_EN in ADX_IN_BYTE_EN0 and ADX_IN_BYTE_EN1, respectively.
4. Byte RAMCTL:

   For programming RAM with the HW_ADR_EN bit cleared in the ADX_AUDIORAMCTL_ADX_CTRL register, set the RAM offset in the RAM_ADR field of the ADX_AUDIORAMCTL_ADX_CTRL register, and set data in the DATA field in the ADX_AUDIORAMCTL_ADX_DATA register.

   For programming RAM with the HW_ADR_EN bit set in the ADX_AUDIORAMCTL_ADX_CTRL register, set the RAM offset in the RAM_ADR field of the ADX_AUDIORAMCTL_ADX_CTRL register to 0, and set data in DATA field in the ADX_AUDIORAMCTL_ADX_DATA register. Hardware auto-increments RAM_ADR after each write.

20.9 Audio Flow Controller (AFC)

The audio flow controller (AFC) can control the flow of traffic between two I2S interfaces running at different clocks that are up to 100ppm apart.

The AFC implements high fidelity interpolation and decimation algorithms that compensate for clock differences.

It can control traffic flow anywhere in the audio route, even to the inputs of AMX blocks.

The AFC can handle burst traffic from DAMs.

20.9.1 AFC Programming Guidelines

1. Program the AFC_AUDIOCIF_AFCTX_CTRL_0 and AFC_AUDIOCIF_AFCRX_CTRL_0 registers according to the programming guidelines given in the ACIF Programming Guidelines subsection.
2. Program the START_THRESHOLD, HIGH_THRESHOLD, and LOW_THRESHOLD registers depending on the use case as shown in the following table.

Note: In the table, the numbers in parentheses are example values.

Use Case	**AFC_THRESHOLDS_I2S_0**			**AFC_THRESHOLDS_AFC_0**			**Destination I2S FIFO Threshold (To be Programmed in I2S_AUDIOCIF_I2SRX_CTRL_0. FIFO_THRESHOLD)**
Actual Value (Typical Value)	HIGH_ THRESHOLD	START_ THRESHOLD	LOW_ THRESHOLD	HIGH_ THRESHOLD	START_ THRESHOLD	LOW_ THRESHOLD	
I2S → AFC → I2S	START_ THRESHOLD+ 1 (8)	START_ THRESHOLD (7)	START_ THRESHOLD - 1 (6)	START_ THRESHOLD + 1 (8)	START_ THRESHOLD (7)	START_ THRESHOLD- 1 (6)	START_THRESHOLD + 1 (AFC_THRESHOLDS_ I2S_0) (8)
I2S → DAM → AFC → I2S	2*SRC-BURST + 1	1*SRC-BURST + 2	1*SRC-BURST	1*SRC-BURST + 1	1*SRC-BURST + 1	1*SRC-BURST + 1	START_THRESHOLD + 1 (AFC_THRESHOLDS_ I2S_0)
I2S → AFC → AMX → I2S	START_ THRESHOLD+ 1 (8)	START_ THRESHOLD (7)	START_ THRESHOLD - 1 (6)	START_ THRESHOLD + 1 (8)	START_ THRESHOLD (7)	START_ THRESHOLD- 1 (6)	Start Threshold + 1 (AFC_THRESHOLDS_ I2S_0) (8)
I2S → DAM → AFC → AMX → I2S	2*SRC-BURST +1	1*SRC-BURST +2	1*SRC-BURST	2*SRC-BURST + 4	1*SRC-BURST + 4	(4)	START_THRESHOLD + 1 (AFC_THRESHOLDS_ I2S_0)

20.10 Audio Hub Registers

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

20.10.1 Audio Crossbar Registers

The following macro defines a register for each RX port. No more than one field can be 1 in a register because an RX port can receive from only one TX port.

Example:

```
reg DAM0_RX0 incr4
0:0         DAM0_TX0          rw        i=0
1:1         I2S0_TX0          rw        i=0
2:2         I2S1_TX0          rw        i=0
;
```

20.10.1.1 AUDIO_APBIF_RXn_0

There are 10 Audio APBIF RX Registers, one per RX port (n = 0 through 9).

For n = 0 through 3: Offset: 0x0 + (n * 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

For n = 4 through 9: Offset: 0x44 + (n * 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
9	0x0	DAM0_TX0 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0 0 = DISABLE 1 = ENABLE

20.10.1.2 AUDIO_I2Sn_RX0_0

There are 5 Audio I2S RX0 Registers, one per I2S channel (n = 0 through 4).

Offset: 0x10 + (n * 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0 0 = DISABLE 1 = ENABLE

20.10.1.3 AUDIO_DAMn_RX0_0

There are 3 Audio DAMn RX0 Registers for the RX0 port (n = 0 through 2).

Offset: 0x24 + (n * 0x8) | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
11	0x0	DAM2_TX0 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0 0 = DISABLE 1 = ENABLE

20.10.1.4 AUDIO_DAMn_RX1_0

There are 3 Audio DAMn RX1 Registers for the RX1 port (n = 0 through 2).

Offset: 0x28 + (n * 0x08) | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1:

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	0x0	I2S0_TX0 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0 0 = DISABLE 1 = ENABLE

20.10.1.5 AUDIO_SPDIF_RXn_0

There are 2 Audio SPDIF RX Registers, one per RX port (n = 0 through 1).

Offset: 0x3c + (n * 0x04) | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4:

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

20.10.1.6 AUDIO_AMX0_RXn_0

There are 4 Audio AMX0 RX Registers, one per RX port (n = 0 through 3).

Offset: 0x5c + (n * 0x04) │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

20.10.1.7 AUDIO_ADX0_RX0_0

Offset: 0x6c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

20.10.1.8 AUDIO_BBC1_RXn_0

There are 2 Audio BBC1 RX registers, one per RX port (n = 0 through 1).

Offset: 0x70 + (n * 0x04) | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

20.10.1.9 AUDIO_AMX1_RXn_0

There are 4 Audio AMX1 RX Registers, one per RX port (n = 0 through 3).

Offset: 0x78 + (n * 0x04) | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

20.10.1.10 AUDIO_ADX1_RX0_0

Offset: 0x88 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

20.10.1.11 AUDIO_AFCn_RX0_0

There are 6 Audio AFCn RX Registers for the RX0 port (n = 0 through 5).

Offset: 0x8c + (n * 0x04) | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

20.10.2 APBIF Registers

20.10.2.1 APBIF_CHANNELn_CTRL_0

There are 4 APBIF Channel Control Registers, one per channel (n = 0 through 3).

Offset: 0x0 + (n * 0x20) | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxx00000000000000000x000x000)

Bit	Reset	Description
31	0x0	TX_ENABLE: 0 = DISABLE 1 = ENABLE
30	0x0	RX_ENABLE: 0 = DISABLE 1 = ENABLE
29	0x0	LOOPBACK: 0 = DISABLE 1 = ENABLE
23:16	0x0	TX_THRESHOLD
15:8	0x0	RX_THRESHOLD
6	0x0	TX_PACK_EN: 0 = DISABLE 1 = ENABLE
5:4	0x0	TX_PACK: 0 = NOP 1 = RSVD 2 = PACK8_4 3 = PACK16
2	0x0	RX_PACK_EN: 0 = DISABLE 1 = ENABLE
1:0	0x0	RX_PACK: 0 = NOP 1 = RSVD 2 = PACK8_4 3 = PACK16

20.10.2.2 APBIF_CHANNELn_CLEAR_0

There are 4 APBIF Channel Clear Registers, one per channel (n = 0 through 3).

Offset: 0x4 + (n * 0x20) | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	TX_SOFT_RESET: 0 = DISABLE 1 = ENABLE
30	0x0	RX_SOFT_RESET: 0 = DISABLE 1 = ENABLE

20.10.2.3 APBIF_CHANNELn_STATUS_0

There are 4 APBIF Channel Status Registers, one per channel (n = 0 through 3).

Offset: 0x8 + (n * 0x20) | Read/Write: RO | Reset: 0xXXXX000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	TX_FREE_COUNT
23:16	X	RX_FREE_COUNT
1	X	TX_TRIG: 0 = DISABLE 1 = ENABLE
0	X	RX_TRIG: 0 = DISABLE 1 = ENABLE

20.10.2.4 APBIF_CHANNELn_TXFIFO_0

There are 4 APBIF Channel TX FIFO Registers, one per channel (n = 0 through 3).

Offset: 0xc + (n * 0x20) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

20.10.2.5 APBIF_CHANNELn_RXFIFO_0

There are 4 APBIF Channel RX FIFO Registers, one per channel (n = 0 through 3).

Offset: 0x10 + (n * 0x20) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

20.10.2.6 APBIF_AUDIOCIF_TXn_CTRL_0

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 the replicate bit is set then previous frame is repeated.

In Tegra K1 devices, only replicating for mono and stereo frames is supported due to the FIFO size limitation. If there are more than 2 channels, the samples in the FIFO are used to fill other multi-channels.

DISABLE -> Disable replication. All zeros will be used.

ENABLE -> Previous frame is repeated.

STEREO_CONV : Convert stereo to mono

CH0 -> pick the first channel

CH1 -> pick the second channel

AVG -> (ch0 + ch1) / 2

valid only in 2 channel to 1 channel conversion

EXPAND : Expand the additional bits with the following method

ZERO -> zero out the expanded field

ONE -> fill the expanded field with 1's

LSFR -> fill the expanded field with a random number

CLIENT_BITS : The number of bits in a sample in the client side

AUDIO_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

AUDIO_CHANNELS : The number of channels in the AUDIO side

the actual channels = AUDIO_CHANNELS + 1, i.e., 0 -> 1 channel

FIFO_THRESHOLD : This specifies the number of words that need to be present in the FIFO before a CIF starts transfers.

0 : start transfer when 1 word is received.

1 : start transfer when 2 words are received.

i : start transfer when i words are received.

For example, if FIFO_THRESHOLD is set to 3, then RxCIF gives rd_req to the I2S/ 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. For all other cases, the default value of zero is the appropriate setting.There are 4 APBIF Audio CIF TX FIFO Registers, one per channel (n = 0 through 3).

Offset: 0x14 + (n * 0x20) | Read/Write: R/W | Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2

Bit	R/W	Reset	Description
			2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
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

20.10.2.7 APBIF_AUDIOCIF_RXn_CTRL_0

There are 4 APBIF Audio CIF RX FIFO Registers, one per channel (n = 0 through 3).

Offset: 0x18 + (n * 0x20) | Read/Write: R/W | Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD

Bit	R/W	Reset	Description
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

20.10.2.8 APBIF_CONFIG_LINK_CTRL_0

Offset: 0x80 | Read/Write: R/W | Reset: 0xX8008000 (0bxxxx10000000000010000000000x000)

Bit	R/W	Reset	Description
31:28	RO	X	MASTER_FIFO_FULL_CNT: Config link master FIFO full count
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 for these 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 counter.(Write-only) 0 = DISABLE 1 = ENABLE
0	RW	0x0	SOFT_RESET: Resets config_link master and slaves.(Write-only) 0 = DISABLE 1 = ENABLE

20.10.2.9 APBIF_MISC_CTRL_0

Offset: 0x84 | Read/Write: R/W | Reset: 0xX0000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0xxx00000)

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

20.10.2.10 APBIF_APBDMA_LIVE_STATUS_0

Offset: 0x88 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description

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: 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

Bit	Reset	Description
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
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

20.10.2.11 APBIF_I2S_LIVE_STATUS_0

Offset: 0x8c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29	X	I2S4_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
28	X	I2S4_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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
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

Bit	Reset	Description
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
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

20.10.2.12 APBIF_DAM0_LIVE_STATUS_0

Offset: 0x90 | Read/Write: RO | Reset: 0x0X00XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
26	X	DAM0_TX_ENABLED: 0 = DISABLE 1 = ENABLE
25	X	DAM0_RX1_ENABLED: 0 = DISABLE 1 = ENABLE
24	X	DAM0_RX0_ENABLED: 0 = DISABLE 1 = ENABLE
15	X	DAM0_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
9	X	DAM0_RX1_FIFO_FULL: 0 = DISABLE 1 = ENABLE
8	X	DAM0_RX0_FIFO_FULL: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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

20.10.2.13 APBIF_DAM1_LIVE_STATUS_0

Offset: 0x98 | Read/Write: RO | Reset: 0x0X00XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
26	X	DAM1_TX_ENABLED: 0 = DISABLE 1 = ENABLE
25	X	DAM1_RX1_ENABLED: 0 = DISABLE 1 = ENABLE
24	X	DAM1_RX0_ENABLED: 0 = DISABLE 1 = ENABLE
15	X	DAM1_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
9	X	DAM1_RX1_FIFO_FULL: 0 = DISABLE 1 = ENABLE
8	X	DAM1_RX0_FIFO_FULL: 0 = DISABLE 1 = ENABLE
7	X	DAM1_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
1	X	DAM1_RX1_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	DAM1_RX0_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

20.10.2.14 APBIF_DAM2_LIVE_STATUS_0

Offset: 0xa0 | Read/Write: RO | Reset: 0x0X00XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
26	X	DAM2_TX_ENABLED: 0 = DISABLE 1 = ENABLE
25	X	DAM2_RX1_ENABLED: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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
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

20.10.2.15 APBIF_SPDIF_LIVE_STATUS_0

Offset: 0xa8 | Read/Write: RO | Reset: 0x000X0XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18	X	SPDIF_TXC_BSY 0 = DISABLE 1 = ENABLE
11	X	SPDIF_USER_TX_ENABLED: 0 = DISABLE 1 = ENABLE
10	X	SPDIF_USER_RX_ENABLED: 0 = DISABLE 1 = ENABLE
9	X	SPDIF_DATA_TX_ENABLED: 0 = DISABLE 1 = ENABLE
8	X	SPDIF_DATA_RX_ENABLED: 0 = DISABLE 1 = ENABLE
7	X	SPDIF_USER_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
6	X	SPDIF_USER_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
5	X	SPDIF_DATA_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
4	X	SPDIF_DATA_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
3	X	SPDIF_USER_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
2	X	SPDIF_USER_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
1	X	SPDIF_DATA_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	SPDIF_DATA_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

20.10.2.16 APBIF_I2S_INT_MASK_0

Because the config registers of the AUDIO clients are daisy-chained, their read latency can be high. To avoid unnecessary latency, the clients provide signals to APBI, mostly interrupt signals so that the CPU/AVP can access them promptly.

These registers should be detailed further with AUDIO clients.

Interrupt Mask and Status for AUDIO Clients

Offset: 0xb0 | Read/Write: R/W | Reset: 0x7fff03ff (0bx111111111111111xxxxxx1111111111)

Bit	Reset	Description
30	0x1	I2S4_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
29	0x1	I2S3_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
28	0x1	I2S2_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
27	0x1	I2S1_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
26	0x1	I2S0_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
25	0x1	I2S4_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
24	0x1	I2S4_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
23	0x1	I2S3_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
22	0x1	I2S3_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
21	0x1	I2S2_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	0x1	I2S2_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
19	0x1	I2S1_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
18	0x1	I2S1_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
17	0x1	I2S0_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
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

20.10.2.17 APBIF_DAM_INT_MASK_0

Offset: 0xb4 | Read/Write: R/W | Reset: 0x00939393 (0bxxxxxxxx1xx1xx111xx1xx111xx1xx11)

Bit	Reset	Description
23	0x1	DAM2_CLIP_INTR: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	0x1	DAM2_TX_DONE: 0 = DISABLE 1 = ENABLE
17	0x1	DAM2_RX1_DONE: 0 = DISABLE 1 = ENABLE
16	0x1	DAM2_RX0_DONE: 0 = DISABLE 1 = ENABLE
15	0x1	DAM1_CLIP_INTR: 0 = DISABLE 1 = ENABLE
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

20.10.2.18 APBIF_SPDIF_INT_MASK_0

Offset: 0xbc | Read/Write: R/W | Reset: 0x00008fff (0bxxxxxxxxxxxxxxxx1xxx111111111111)

Bit	Reset	Description
15	0x1	SPDIF_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
11	0x1	SPDIF_BAD_PREAMBLE: 0 = DISABLE 1 = ENABLE
10	0x1	SPDIF_B_SYNC: 0 = DISABLE 1 = ENABLE
9	0x1	SPDIF_RX_CHANNEL: 0 = DISABLE 1 = ENABLE
8	0x1	SPDIF_RX_IU: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	0x1	SPDIF_USER_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x1	SPDIF_USER_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x1	SPDIF_DATA_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
4	0x1	SPDIF_DATA_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
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

20.10.2.19 APBIF_APBIF_INT_MASK_0

Offset: 0xc0 | Read/Write: R/W | Reset: 0x0001ffff (0bxxxxxxxxxxxxxxxx11111111111111111)

Bit	Reset	Description
16	0x1	CONFIG_LINK_TIMEOUT: 0 = DISABLE 1 = ENABLE
15	0x1	APBIF_CH3_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
14	0x1	APBIF_CH3_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
13	0x1	APBIF_CH2_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
12	0x1	APBIF_CH2_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
11	0x1	APBIF_CH1_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
10	0x1	APBIF_CH1_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
9	0x1	APBIF_CH0_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	0x1	APBIF_CH0_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	0x1	APBIF_CH3_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x1	APBIF_CH3_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x1	APBIF_CH2_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
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

20.10.2.20 APBIF_I2S_INT_STATUS_0

Bits in this register are high after the interrupt condition is satisfied. A write to this register clears the bits corresponding to the data bits that are high in the write data.

Interrupt Status Registers

Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0bx00000000000000xxxxxx0000000000)

Bit	Reset	Description
30	0x0	I2S4_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
29	0x0	I2S3_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
28	0x0	I2S2_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
27	0x0	I2S1_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
26	0x0	I2S0_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
25	0x0	I2S4_TXCIF_OVERRUN: 0 = DISABLE

Bit	Reset	Description
		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
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

Bit	Reset	Description
		1 = ENABLE
0	0x0	I2S0_TX_DONE: 0 = DISABLE 1 = ENABLE

20.10.2.21 APBIF_DAM_INT_STATUS_0

Offset: 0xcc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx0xx0xx000xx0xx000xx0xx00)

Bit	Reset	Description
23	0x0	DAM2_CLIP_INTR: 0 = DISABLE 1 = ENABLE
20	0x0	DAM2_TX_DONE: 0 = DISABLE 1 = ENABLE
17	0x0	DAM2_RX1_DONE: 0 = DISABLE 1 = ENABLE
16	0x0	DAM2_RX0_DONE: 0 = DISABLE 1 = ENABLE
15	0x0	DAM1_CLIP_INTR: 0 = DISABLE 1 = ENABLE
12	0x0	DAM1_TX_DONE: 0 = DISABLE 1 = ENABLE
9	0x0	DAM1_RX1_DONE: 0 = DISABLE 1 = ENABLE
8	0x0	DAM1_RX0_DONE: 0 = DISABLE 1 = ENABLE
7	0x0	DAM0_CLIP_INTR: 0 = DISABLE 1 = ENABLE
4	0x0	DAM0_TX_DONE: 0 = DISABLE 1 = ENABLE
1	0x0	DAM0_RX1_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	DAM0_RX0_DONE: 0 = DISABLE 1 = ENABLE

20.10.2.22 APBIF_SPDIF_INT_STATUS_0

Offset: 0xd4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxx000000000000)

Bit	Reset	Description
15	0x0	SPDIF_FLOW_CTL_INT: 0 = DISABLE

Bit	Reset	Description
		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
1	0x0	SPDIF_DATA_TX_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	SPDIF_DATA_RX_DONE: 0 = DISABLE 1 = ENABLE

20.10.2.23 APBIF_APBIF_INT_STATUS_0

Offset: 0xd8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000000000000)

Bit	Reset	Description
16	0x0	CONFIG_LINK_TIMEOUT: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_CH3_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_CH3_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
13	0x0	APBIF_CH2_APBDMA_RX_OVERRUN: 0 = DISABLE

Bit	Reset	Description
		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
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

20.10.2.24 APBIF_I2S_INT_SOURCE_0

Interrupt Source

Offset: 0xe0 | Read/Write: RO | Reset: 0xXXXX0XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	I2S4_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
29	X	I2S3_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE

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

20.10.2.25 APBIF_DAM_INT_SOURCE_0

Offset: 0xe4 | Read/Write: RO | Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23	X	DAM2_CLIP_INTR: 0 = DISABLE 1 = ENABLE
20	X	DAM2_TX_DONE: 0 = DISABLE 1 = ENABLE
17	X	DAM2_RX1_DONE: 0 = DISABLE 1 = ENABLE
16	X	DAM2_RX0_DONE: 0 = DISABLE 1 = ENABLE
15	X	DAM1_CLIP_INTR: 0 = DISABLE 1 = ENABLE
12	X	DAM1_TX_DONE: 0 = DISABLE 1 = ENABLE
9	X	DAM1_RX1_DONE: 0 = DISABLE 1 = ENABLE
8	X	DAM1_RX0_DONE: 0 = DISABLE 1 = ENABLE
7	X	DAM0_CLIP_INTR: 0 = DISABLE 1 = ENABLE
4	X	DAM0_TX_DONE: 0 = DISABLE 1 = ENABLE
1	X	DAM0_RX1_DONE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	X	DAM0_RX0_DONE: 0 = DISABLE 1 = ENABLE

20.10.2.26 APBIF_SPDIF_INT_SOURCE_0

Offset: 0xec | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15	X	SPDIF_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
11	X	SPDIF_BAD_PREAMBLE: 0 = DISABLE 1 = ENABLE
10	X	SPDIF_B_SYNC: 0 = DISABLE 1 = ENABLE
9	X	SPDIF_RX_CHANNEL: 0 = DISABLE 1 = ENABLE
8	X	SPDIF_RX_IU: 0 = DISABLE 1 = ENABLE
7	X	SPDIF_USER_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	X	SPDIF_USER_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	X	SPDIF_DATA_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
4	X	SPDIF_DATA_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	X	SPDIF_USER_TX_DONE: 0 = DISABLE 1 = ENABLE
2	X	SPDIF_USER_RX_DONE: 0 = DISABLE 1 = ENABLE
1	X	SPDIF_DATA_TX_DONE: 0 = DISABLE 1 = ENABLE
0	X	SPDIF_DATA_RX_DONE: 0 = DISABLE 1 = ENABLE

20.10.2.27 APBIF_APBIF_INT_SOURCE_0

Offset: 0xf0 | Read/Write: RO | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
16	X	CONFIG_LINK_TIMEOUT: 0 = DISABLE 1 = ENABLE
15	X	APBIF_CH3_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
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

20.10.2.28 APBIF_I2S_INT_SET_0

Interrupt Set

Offset: 0xf8 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxx0000000000)

Bit	Reset	Description
30	0x0	I2S4_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
29	0x0	I2S3_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
28	0x0	I2S2_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
27	0x0	I2S1_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
26	0x0	I2S0_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
25	0x0	I2S4_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
24	0x0	I2S4_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
23	0x0	I2S3_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
22	0x0	I2S3_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
21	0x0	I2S2_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
20	0x0	I2S2_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
19	0x0	I2S1_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
18	0x0	I2S1_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
17	0x0	I2S0_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
16	0x0	I2S0_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
9	0x0	I2S4_RX_DONE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	0x0	I2S4_TX_DONE: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_RX_DONE: 0 = DISABLE 1 = ENABLE
6	0x0	I2S3_TX_DONE: 0 = DISABLE 1 = ENABLE
5	0x0	I2S2_RX_DONE: 0 = DISABLE 1 = ENABLE
4	0x0	I2S2_TX_DONE: 0 = DISABLE 1 = ENABLE
3	0x0	I2S1_RX_DONE: 0 = DISABLE 1 = ENABLE
2	0x0	I2S1_TX_DONE: 0 = DISABLE 1 = ENABLE
1	0x0	I2S0_RX_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	I2S0_TX_DONE: 0 = DISABLE 1 = ENABLE

20.10.2.29 APBIF_DAM_INT_SET_0

Offset: 0xfc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx0xx0xx000xx0xx000xx0xx00)

Bit	Reset	Description
23	0x0	DAM2_CLIP_INTR: 0 = DISABLE 1 = ENABLE
20	0x0	DAM2_TX_DONE: 0 = DISABLE 1 = ENABLE
17	0x0	DAM2_RX1_DONE: 0 = DISABLE 1 = ENABLE
16	0x0	DAM2_RX0_DONE: 0 = DISABLE 1 = ENABLE
15	0x0	DAM1_CLIP_INTR: 0 = DISABLE 1 = ENABLE
12	0x0	DAM1_TX_DONE: 0 = DISABLE 1 = ENABLE
9	0x0	DAM1_RX1_DONE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	0x0	DAM1_RX0_DONE: 0 = DISABLE 1 = ENABLE
7	0x0	DAM0_CLIP_INTR: 0 = DISABLE 1 = ENABLE
4	0x0	DAM0_TX_DONE: 0 = DISABLE 1 = ENABLE
1	0x0	DAM0_RX1_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	DAM0_RX0_DONE: 0 = DISABLE 1 = ENABLE

20.10.2.30 APBIF_SPDIF_INT_SET_0

Offset: 0x100 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxx000000000000)

Bit	Reset	Description
15	0x0	SPDIF_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
11	0x0	SPDIF_BAD_PREAMBLE: 0 = DISABLE 1 = ENABLE
10	0x0	SPDIF_B_SYNC: 0 = DISABLE 1 = ENABLE
9	0x0	SPDIF_RX_CHANNEL: 0 = DISABLE 1 = ENABLE
8	0x0	SPDIF_RX_IU: 0 = DISABLE 1 = ENABLE
7	0x0	SPDIF_USER_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	SPDIF_USER_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	SPDIF_DATA_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	SPDIF_DATA_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	SPDIF_USER_TX_DONE: 0 = DISABLE 1 = ENABLE
2	0x0	SPDIF_USER_RX_DONE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x0	SPDIF_DATA_TX_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	SPDIF_DATA_RX_DONE: 0 = DISABLE 1 = ENABLE

20.10.2.31 APBIF_APBIF_INT_SET_0

Offset: 0x104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000000000000)

Bit	Reset	Description
16	0x0	CONFIG_LINK_TIMEOUT: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_CH3_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_CH3_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
13	0x0	APBIF_CH2_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
12	0x0	APBIF_CH2_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
11	0x0	APBIF_CH1_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
10	0x0	APBIF_CH1_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
9	0x0	APBIF_CH0_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
8	0x0	APBIF_CH0_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	0x0	APBIF_CH3_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	APBIF_CH3_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	APBIF_CH2_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	APBIF_CH2_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_CH1_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	APBIF_CH1_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_CH0_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_CH0_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

20.10.2.32 APBIF_APBDMA_LIVE_TX_DMA_FIFO_EMPTY_STATUS_0

Offset: 0x10c │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH9_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
8	X	CH8_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
7	X	CH7_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
6	X	CH6_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
5	X	CH5_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
4	X	CH4_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
3	X	CH3_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
2	X	CH2_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
1	X	CH1_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	CH0_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

20.10.2.33 APBIF_APBDMA_LIVE_TX_CIF_FIFO_EMPTY_STATUS_0

Offset: 0x110 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH9_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	X	CH8_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
7	X	CH7_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
6	X	CH6_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
5	X	CH5_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
4	X	CH4_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
3	X	CH3_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
2	X	CH2_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
1	X	CH1_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	CH0_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

20.10.2.34 APBIF_APBDMA_LIVE_RX_DMA_FIFO_EMPTY_STATUS_0

Offset: 0x114 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH9_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
8	X	CH8_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
7	X	CH7_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
6	X	CH6_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
5	X	CH5_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
4	X	CH4_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
3	X	CH3_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	X	CH2_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
1	X	CH1_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	CH0_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

20.10.2.35 APBIF_APBDMA_LIVE_RX_CIF_FIFO_EMPTY_STATUS_0

Offset: 0x118 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH9_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
8	X	CH8_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
7	X	CH7_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
6	X	CH6_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
5	X	CH5_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
4	X	CH4_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
3	X	CH3_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
2	X	CH2_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
1	X	CH1_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	CH0_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

20.10.2.36 APBIF_APBDMA_LIVE_TX_DMA_FIFO_FULL_STATUS_0

Offset: 0x11c │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH9_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	X	CH8_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
7	X	CH7_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
6	X	CH6_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
5	X	CH5_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
4	X	CH4_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
3	X	CH3_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
2	X	CH2_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
1	X	CH1_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
0	X	CH0_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE

20.10.2.37 APBIF_APBDMA_LIVE_TX_CIF_FIFO_FULL_STATUS_0

Offset: 0x120 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH9_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
8	X	CH8_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
7	X	CH7_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
6	X	CH6_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
5	X	CH5_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
4	X	CH4_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
3	X	CH3_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	X	CH2_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
1	X	CH1_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
0	X	CH0_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE

20.10.2.38 APBIF_APBDMA_LIVE_RX_DMA_FIFO_FULL_STATUS_0

Offset: 0x124 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH9_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
8	X	CH8_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
7	X	CH7_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
6	X	CH6_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
5	X	CH5_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
4	X	CH4_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
3	X	CH3_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
2	X	CH2_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
1	X	CH1_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
0	X	CH0_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE

20.10.2.39 APBIF_APBDMA_LIVE_RX_CIF_FIFO_FULL_STATUS_0

Offset: 0x128 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH9_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	X	CH8_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
7	X	CH7_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
6	X	CH6_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
5	X	CH5_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
4	X	CH4_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
3	X	CH3_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
2	X	CH2_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
1	X	CH1_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
0	X	CH0_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE

20.10.2.40 APBIF_APBIF_RXCIF_UNDERRUN_INT_MASK_0

Offset: 0x12c │ Read/Write: R/W │ Reset: 0x000003ff (0bxxxxxxxxxxxxxxxxxxxxxxxx1111111111)

Bit	Reset	Description
9	0x1	APBIF_CH9_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
8	0x1	APBIF_CH8_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	0x1	APBIF_CH7_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
6	0x1	APBIF_CH6_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x1	APBIF_CH5_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
4	0x1	APBIF_CH4_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	0x1	APBIF_CH3_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x1	APBIF_CH2_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	0x1	APBIF_CH1_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
0	0x1	APBIF_CH0_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

20.10.2.41 APBIF_APBIF_TXCIF_OVERRUN_INT_MASK_0

Offset: 0x130 │ Read/Write: R/W │ Reset: 0x000003ff (0bxxxxxxxxxxxxxxxxxxxxxxxx1111111111)

Bit	Reset	Description
9	0x1	APBIF_CH9_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
8	0x1	APBIF_CH8_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
7	0x1	APBIF_CH7_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x1	APBIF_CH6_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
5	0x1	APBIF_CH5_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
4	0x1	APBIF_CH4_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
3	0x1	APBIF_CH3_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
2	0x1	APBIF_CH2_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
1	0x1	APBIF_CH1_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
0	0x1	APBIF_CH0_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE

20.10.2.42 APBIF_APBIF_APBDMA_UNDERRUN_INT_MASK_0

Offset: 0x134 │ Read/Write: R/W │ Reset: 0x000003ff (0bxxxxxxxxxxxxxxxxxxxxxxxx1111111111)

Bit	Reset	Description
9	0x1	APBIF_CH9_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	0x1	APBIF_CH8_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	0x1	APBIF_CH7_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
6	0x1	APBIF_CH6_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x1	APBIF_CH5_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
4	0x1	APBIF_CH4_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	0x1	APBIF_CH3_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
2	0x1	APBIF_CH2_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	0x1	APBIF_CH1_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
0	0x1	APBIF_CH0_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE

20.10.2.43 APBIF_APBIF_APBDMA_OVERRUN_INT_MASK_0

Offset: 0x138 │ Read/Write: R/W │ Reset: 0x000003ff (0bxxxxxxxxxxxxxxxxxxxxxxxx1111111111)

Bit	Reset	Description
9	0x1	APBIF_CH9_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
8	0x1	APBIF_CH8_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
7	0x1	APBIF_CH7_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x1	APBIF_CH6_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
5	0x1	APBIF_CH5_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
4	0x1	APBIF_CH4_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
3	0x1	APBIF_CH3_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x1	APBIF_CH2_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
1	0x1	APBIF_CH1_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
0	0x1	APBIF_CH0_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE

20.10.2.44 APBIF_APBIF_RXCIF_UNDERRUN_INT_STATUS_0

Offset: 0x13c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	0x0	APBIF_CH9_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
8	0x0	APBIF_CH8_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	0x0	APBIF_CH7_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	APBIF_CH6_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	APBIF_CH5_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	APBIF_CH4_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_CH3_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_CH2_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_CH1_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_CH0_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

20.10.2.45 APBIF_APBIF_TXCIF_OVERRUN_INT_STATUS_0

Offset: 0x140 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	0x0	APBIF_CH9_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	0x0	APBIF_CH8_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
7	0x0	APBIF_CH7_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	APBIF_CH6_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	APBIF_CH5_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	APBIF_CH4_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_CH3_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_CH2_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_CH1_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_CH0_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE

20.10.2.46 APBIF_APBIF_APBDMA_UNDERRUN_INT_STATUS_0

Offset: 0x144 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	0x0	APBIF_CH9_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
8	0x0	APBIF_CH8_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	0x0	APBIF_CH7_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	APBIF_CH6_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	APBIF_CH5_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	APBIF_CH4_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_CH3_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	APBIF_CH2_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_CH1_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_CH0_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE

20.10.2.47 APBIF_APBIF_APBDMA_OVERRUN_INT_STATUS_0

Offset: 0x148 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	0x0	APBIF_CH9_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
8	0x0	APBIF_CH8_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
7	0x0	APBIF_CH7_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	APBIF_CH6_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	APBIF_CH5_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	APBIF_CH4_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_CH3_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_CH2_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_CH1_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_CH0_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE

20.10.2.48 APBIF_APBIF_RXCIF_UNDERRUN_INT_SOURCE_0

Offset: 0x14c │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	APBIF_CH9_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	X	APBIF_CH8_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	X	APBIF_CH7_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
6	X	APBIF_CH6_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	X	APBIF_CH5_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
4	X	APBIF_CH4_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	X	APBIF_CH3_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
2	X	APBIF_CH2_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	X	APBIF_CH1_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
0	X	APBIF_CH0_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

20.10.2.49 APBIF_APBIF_TXCIF_OVERRUN_INT_SOURCE_0

Offset: 0x150 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	APBIF_CH9_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
8	X	APBIF_CH8_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
7	X	APBIF_CH7_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	X	APBIF_CH6_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
5	X	APBIF_CH5_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
4	X	APBIF_CH4_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
3	X	APBIF_CH3_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	X	APBIF_CH2_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
1	X	APBIF_CH1_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
0	X	APBIF_CH0_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE

20.10.2.50 APBIF_APBIF_APBDMA_UNDERRUN_INT_SOURCE_0

Offset: 0x154 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	APBIF_CH9_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
8	X	APBIF_CH8_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	X	APBIF_CH7_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
6	X	APBIF_CH6_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	X	APBIF_CH5_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
4	X	APBIF_CH4_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	X	APBIF_CH3_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
2	X	APBIF_CH2_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	X	APBIF_CH1_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
0	X	APBIF_CH0_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE

20.10.2.51 APBIF_APBIF_APBDMA_OVERRUN_INT_SOURCE_0

Offset: 0x158 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	APBIF_CH9_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	X	APBIF_CH8_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
7	X	APBIF_CH7_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
6	X	APBIF_CH6_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
5	X	APBIF_CH5_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
4	X	APBIF_CH4_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
3	X	APBIF_CH3_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
2	X	APBIF_CH2_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
1	X	APBIF_CH1_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
0	X	APBIF_CH0_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE

20.10.2.52 APBIF_APBIF_RXCIF_UNDERRUN_INT_SET_0

Offset: 0x15c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	0x0	APBIF_CH9_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
8	0x0	APBIF_CH8_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	0x0	APBIF_CH7_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	APBIF_CH6_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	APBIF_CH5_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	APBIF_CH4_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_CH3_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	APBIF_CH2_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_CH1_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_CH0_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

20.10.2.53 APBIF_APBIF_TXCIF_OVERRUN_INT_SET_0

Offset: 0x160 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	0x0	APBIF_CH9_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
8	0x0	APBIF_CH8_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
7	0x0	APBIF_CH7_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	APBIF_CH6_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	APBIF_CH5_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	APBIF_CH4_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_CH3_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_CH2_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_CH1_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_CH0_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE

20.10.2.54 APBIF_APBIF_APBDMA_UNDERRUN_INT_SET_0

Offset: 0x164 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	0x0	APBIF_CH9_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	0x0	APBIF_CH8_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	0x0	APBIF_CH7_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	APBIF_CH6_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	APBIF_CH5_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	APBIF_CH4_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_CH3_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_CH2_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_CH1_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_CH0_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE

20.10.2.55 APBIF_APBIF_APBDMA_OVERRUN_INT_SET_0

Offset: 0x168 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	0x0	APBIF_CH9_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
8	0x0	APBIF_CH8_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
7	0x0	APBIF_CH7_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	APBIF_CH6_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	APBIF_CH5_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	APBIF_CH4_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_CH3_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	APBIF_CH2_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_CH1_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_CH0_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE

20.10.2.56 APBIF_AMX0_LIVE_STATUS_0

Offset: 0x178 │ Read/Write: RO │ Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23	X	AMX0_CH3_RX_ENABLED: 0 = DISABLE 1 = ENABLE
22	X	AMX0_CH2_RX_ENABLED: 0 = DISABLE 1 = ENABLE
21	X	AMX0_CH1_RX_ENABLED: 0 = DISABLE 1 = ENABLE
20	X	AMX0_CH0_RX_ENABLED: 0 = DISABLE 1 = ENABLE
16	X	AMX0_TX_ENABLED: 0 = DISABLE 1 = ENABLE
15	X	AMX0_CH3_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
14	X	AMX0_CH2_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
13	X	AMX0_CH1_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
12	X	AMX0_CH0_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
8	X	AMX0_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
7	X	AMX0_CH3_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
6	X	AMX0_CH2_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
5	X	AMX0_CH1_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	X	AMX0_CH0_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	AMX0_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

20.10.2.57 APBIF_AMX1_LIVE_STATUS_0

Offset: 0x17c │ Read/Write: RO │ Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23	X	AMX1_CH3_RX_ENABLED: 0 = DISABLE 1 = ENABLE
22	X	AMX1_CH2_RX_ENABLED: 0 = DISABLE 1 = ENABLE
21	X	AMX1_CH1_RX_ENABLED: 0 = DISABLE 1 = ENABLE
20	X	AMX1_CH0_RX_ENABLED: 0 = DISABLE 1 = ENABLE
16	X	AMX1_TX_ENABLED: 0 = DISABLE 1 = ENABLE
15	X	AMX1_CH3_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
14	X	AMX1_CH2_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
13	X	AMX1_CH1_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
12	X	AMX1_CH0_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
8	X	AMX1_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
7	X	AMX1_CH3_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
6	X	AMX1_CH2_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
5	X	AMX1_CH1_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
4	X	AMX1_CH0_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	X	AMX1_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

20.10.2.58 APBIF_ADX0_LIVE_STATUS_0

Offset: 0x180 │ Read/Write: RO │ Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23	X	ADX0_CH3_TX_ENABLED: 0 = DISABLE 1 = ENABLE
22	X	ADX0_CH2_TX_ENABLED: 0 = DISABLE 1 = ENABLE
21	X	ADX0_CH1_TX_ENABLED: 0 = DISABLE 1 = ENABLE
20	X	ADX0_CH0_TX_ENABLED: 0 = DISABLE 1 = ENABLE
16	X	ADX0_RX_ENABLED: 0 = DISABLE 1 = ENABLE
15	X	ADX0_CH3_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
14	X	ADX0_CH2_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
13	X	ADX0_CH1_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
12	X	ADX0_CH0_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
8	X	ADX0_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
7	X	ADX0_CH3_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
6	X	ADX0_CH2_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
5	X	ADX0_CH1_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
4	X	ADX0_CH0_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	ADX0_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

20.10.2.59 APBIF_ADX1_LIVE_STATUS_0

Offset: 0x184 │ Read/Write: RO │ Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23	X	ADX1_CH3_TX_ENABLED: 0 = DISABLE 1 = ENABLE
22	X	ADX1_CH2_TX_ENABLED: 0 = DISABLE 1 = ENABLE
21	X	ADX1_CH1_TX_ENABLED: 0 = DISABLE 1 = ENABLE
20	X	ADX1_CH0_TX_ENABLED: 0 = DISABLE 1 = ENABLE
16	X	ADX1_RX_ENABLED: 0 = DISABLE 1 = ENABLE
15	X	ADX1_CH3_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
14	X	ADX1_CH2_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
13	X	ADX1_CH1_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
12	X	ADX1_CH0_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
8	X	ADX1_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
7	X	ADX1_CH3_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
6	X	ADX1_CH2_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
5	X	ADX1_CH1_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
4	X	ADX1_CH0_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	ADX1_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

20.10.2.60 APBIF_AMX_INT_MASK_0

Offset: 0x188 | Read/Write: R/W | Reset: 0x0000f1f1 (0bxxxxxxxxxxxxxxxx1111xxx11111xxx1)

Bit	Reset	Description
15	0x1	AMX1_CH3_RX_DONE: 0 = DISABLE 1 = ENABLE
14	0x1	AMX1_CH2_RX_DONE: 0 = DISABLE 1 = ENABLE
13	0x1	AMX1_CH1_RX_DONE: 0 = DISABLE 1 = ENABLE
12	0x1	AMX1_CH0_RX_DONE: 0 = DISABLE 1 = ENABLE
8	0x1	AMX1_TX_DONE: 0 = DISABLE 1 = ENABLE
7	0x1	AMX0_CH3_RX_DONE: 0 = DISABLE 1 = ENABLE
6	0x1	AMX0_CH2_RX_DONE: 0 = DISABLE 1 = ENABLE
5	0x1	AMX0_CH1_RX_DONE: 0 = DISABLE 1 = ENABLE
4	0x1	AMX0_CH0_RX_DONE: 0 = DISABLE 1 = ENABLE
0	0x1	AMX0_TX_DONE: 0 = DISABLE 1 = ENABLE

20.10.2.61 APBIF_ADX_INT_MASK_0

Offset: 0x190 | Read/Write: R/W | Reset: 0x0000f1f1 (0bxxxxxxxxxxxxxxxx1111xxx11111xxx1)

Bit	Reset	Description
15	0x1	ADX1_CH3_TX_DONE: 0 = DISABLE 1 = ENABLE
14	0x1	ADX1_CH2_TX_DONE: 0 = DISABLE 1 = ENABLE
13	0x1	ADX1_CH1_TX_DONE: 0 = DISABLE 1 = ENABLE
12	0x1	ADX1_CH0_TX_DONE: 0 = DISABLE 1 = ENABLE
8	0x1	ADX1_RX_DONE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	0x1	ADX0_CH3_TX_DONE: 0 = DISABLE 1 = ENABLE
6	0x1	ADX0_CH2_TX_DONE: 0 = DISABLE 1 = ENABLE
5	0x1	ADX0_CH1_TX_DONE: 0 = DISABLE 1 = ENABLE
4	0x1	ADX0_CH0_TX_DONE: 0 = DISABLE 1 = ENABLE
0	0x1	ADX0_RX_DONE: 0 = DISABLE 1 = ENABLE

20.10.2.62 APBIF_AMX_INT_STATUS_0

Offset: 0x198 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx0000xxx00000xxx0)

Bit	Reset	Description
15	0x0	AMX1_CH3_RX_DONE: 0 = DISABLE 1 = ENABLE
14	0x0	AMX1_CH2_RX_DONE: 0 = DISABLE 1 = ENABLE
13	0x0	AMX1_CH1_RX_DONE: 0 = DISABLE 1 = ENABLE
12	0x0	AMX1_CH0_RX_DONE: 0 = DISABLE 1 = ENABLE
8	0x0	AMX1_TX_DONE: 0 = DISABLE 1 = ENABLE
7	0x0	AMX0_CH3_RX_DONE: 0 = DISABLE 1 = ENABLE
6	0x0	AMX0_CH2_RX_DONE: 0 = DISABLE 1 = ENABLE
5	0x0	AMX0_CH1_RX_DONE: 0 = DISABLE 1 = ENABLE
4	0x0	AMX0_CH0_RX_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	AMX0_TX_DONE: 0 = DISABLE 1 = ENABLE

20.10.2.63 APBIF_ADX_INT_STATUS_0

Offset: 0x1a0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx0000xxx00000xxx0)

Bit	Reset	Description
15	0x0	ADX1_CH3_TX_DONE: 0 = DISABLE 1 = ENABLE
14	0x0	ADX1_CH2_TX_DONE: 0 = DISABLE 1 = ENABLE
13	0x0	ADX1_CH1_TX_DONE: 0 = DISABLE 1 = ENABLE
12	0x0	ADX1_CH0_TX_DONE: 0 = DISABLE 1 = ENABLE
8	0x0	ADX1_RX_DONE: 0 = DISABLE 1 = ENABLE
7	0x0	ADX0_CH3_TX_DONE: 0 = DISABLE 1 = ENABLE
6	0x0	ADX0_CH2_TX_DONE: 0 = DISABLE 1 = ENABLE
5	0x0	ADX0_CH1_TX_DONE: 0 = DISABLE 1 = ENABLE
4	0x0	ADX0_CH0_TX_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	ADX0_RX_DONE: 0 = DISABLE 1 = ENABLE

20.10.2.64 APBIF_AMX_INT_SOURCE_0

Offset: 0x1a8 │ Read/Write: RO │ Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15	X	AMX1_CH3_RX_DONE: 0 = DISABLE 1 = ENABLE
14	X	AMX1_CH2_RX_DONE: 0 = DISABLE 1 = ENABLE
13	X	AMX1_CH1_RX_DONE: 0 = DISABLE 1 = ENABLE
12	X	AMX1_CH0_RX_DONE: 0 = DISABLE 1 = ENABLE
8	X	AMX1_TX_DONE: 0 = DISABLE 1 = ENABLE
7	X	AMX0_CH3_RX_DONE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	X	AMX0_CH2_RX_DONE: 0 = DISABLE 1 = ENABLE
5	X	AMX0_CH1_RX_DONE: 0 = DISABLE 1 = ENABLE
4	X	AMX0_CH0_RX_DONE: 0 = DISABLE 1 = ENABLE
0	X	AMX0_TX_DONE: 0 = DISABLE 1 = ENABLE

20.10.2.65 APBIF_ADX_INT_SOURCE_0

Offset: 0x1b0 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15	X	ADX1_CH3_TX_DONE: 0 = DISABLE 1 = ENABLE
14	X	ADX1_CH2_TX_DONE: 0 = DISABLE 1 = ENABLE
13	X	ADX1_CH1_TX_DONE: 0 = DISABLE 1 = ENABLE
12	X	ADX1_CH0_TX_DONE: 0 = DISABLE 1 = ENABLE
8	X	ADX1_RX_DONE: 0 = DISABLE 1 = ENABLE
7	X	ADX0_CH3_TX_DONE: 0 = DISABLE 1 = ENABLE
6	X	ADX0_CH2_TX_DONE: 0 = DISABLE 1 = ENABLE
5	X	ADX0_CH1_TX_DONE: 0 = DISABLE 1 = ENABLE
4	X	ADX0_CH0_TX_DONE: 0 = DISABLE 1 = ENABLE
0	X	ADX0_RX_DONE: 0 = DISABLE 1 = ENABLE

20.10.2.66 APBIF_AMX_INT_SET_0

Offset: 0x1b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000xxx00000xxx0)

Bit	Reset	Description
15	0x0	AMX1_CH3_RX_DONE: 0 = DISABLE 1 = ENABLE
14	0x0	AMX1_CH2_RX_DONE: 0 = DISABLE 1 = ENABLE
13	0x0	AMX1_CH1_RX_DONE: 0 = DISABLE 1 = ENABLE
12	0x0	AMX1_CH0_RX_DONE: 0 = DISABLE 1 = ENABLE
8	0x0	AMX1_TX_DONE: 0 = DISABLE 1 = ENABLE
7	0x0	AMX0_CH3_RX_DONE: 0 = DISABLE 1 = ENABLE
6	0x0	AMX0_CH2_RX_DONE: 0 = DISABLE 1 = ENABLE
5	0x0	AMX0_CH1_RX_DONE: 0 = DISABLE 1 = ENABLE
4	0x0	AMX0_CH0_RX_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	AMX0_TX_DONE: 0 = DISABLE 1 = ENABLE

20.10.2.67 APBIF_ADX_INT_SET_0

Offset: 0x1c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000xxx00000xxx0)

Bit	Reset	Description
15	0x0	ADX1_CH3_TX_DONE: 0 = DISABLE 1 = ENABLE
14	0x0	ADX1_CH2_TX_DONE: 0 = DISABLE 1 = ENABLE
13	0x0	ADX1_CH1_TX_DONE: 0 = DISABLE 1 = ENABLE
12	0x0	ADX1_CH0_TX_DONE: 0 = DISABLE 1 = ENABLE
8	0x0	ADX1_RX_DONE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	0x0	ADX0_CH3_TX_DONE: 0 = DISABLE 1 = ENABLE
6	0x0	ADX0_CH2_TX_DONE: 0 = DISABLE 1 = ENABLE
5	0x0	ADX0_CH1_TX_DONE: 0 = DISABLE 1 = ENABLE
4	0x0	ADX0_CH0_TX_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	ADX0_RX_DONE: 0 = DISABLE 1 = ENABLE

20.10.2.68 APBIF_MISC_LIVE_STATUS0_0

Offset: 0x1d0 │ Read/Write: RO │ Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
12	X	I2S4_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
11	X	I2S3_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
10	X	I2S2_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
9	X	I2S1_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
8	X	I2S0_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
4	X	I2S4_CLKEN: 0 = DISABLE 1 = ENABLE
3	X	I2S3_CLKEN: 0 = DISABLE 1 = ENABLE
2	X	I2S2_CLKEN: 0 = DISABLE 1 = ENABLE
1	X	I2S1_CLKEN: 0 = DISABLE 1 = ENABLE
0	X	I2S0_CLKEN: 0 = DISABLE 1 = ENABLE

20.10.2.69 APBIF_MISC_LIVE_STATUS1_0

Offset: 0x1d4 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11	X	AMX1_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
10	X	AMX1_CLKEN: 0 = DISABLE 1 = ENABLE
9	X	AMX0_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
8	X	AMX0_CLKEN: 0 = DISABLE 1 = ENABLE
5	X	ADX1_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
4	X	ADX1_CLKEN: 0 = DISABLE 1 = ENABLE
1	X	ADX0_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
0	X	ADX0_CLKEN: 0 = DISABLE 1 = ENABLE

20.10.2.70 APBIF_MISC_LIVE_STATUS2_0

Offset: 0x1d8 │ Read/Write: RO │ Reset: 0x00000X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10	X	DAM2_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
9	X	DAM1_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
8	X	DAM0_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
2	X	DAM2_CLKEN: 0 = DISABLE 1 = ENABLE
1	X	DAM1_CLKEN: 0 = DISABLE 1 = ENABLE
0	X	DAM0_CLKEN: 0 = DISABLE 1 = ENABLE

20.10.2.71 APBIF_MISC_LIVE_STATUS3_0

Offset: 0x1dc | Read/Write: RO | Reset: 0x00000X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	AUDIO_CFG_CLKEN: 0 = DISABLE 1 = ENABLE
8	X	APBIF_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
2	X	SPDIF_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
1	X	SPDIF_OUT_CLKEN: 0 = DISABLE 1 = ENABLE
0	X	SPDIF_IN_CLKEN: 0 = DISABLE 1 = ENABLE

20.10.3 APBIF2 Registers

Audio channels 4 through 9 have their own set of 7 APBIF2 registers. Each channel has 32 bytes of address space. This subsection defines one complete set of APBIF2 registers. The register spaces per APBIF2 channel are listed in the table below.

Table 73: APBIF2 Channel Register Mapping

Register Space	Channel Registers
0x0 – 0x1f	Channel 4 APBIF2 Channel Registers
0x20 – 0x3f	Channel 5 APBIF2 Channel Registers
0x40 – 0x5f	Channel 6 APBIF2 Channel Registers
0x60 – 0x7f	Channel 7 APBIF2 Channel Registers
0x80 – 0x9f	Channel 8 APBIF2 Channel Registers
0xa0 – 0x17f	Channel 9 APBIF2 Channel Registers

20.10.3.1 APBIF2_CHANNELn_CTRL_0

There are 6 APBIF2 Channel Control Registers, one per channel (n = 4 through 9).

Offset: 0x0 + (n * 0x20) | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxx00000000000000000x000x000)

Bit	Reset	Description
31	0x0	TX_ENABLE: Enable TX channel to AUDIO. 0 = DISABLE 1 = ENABLE
30	0x0	RX_ENABLE: Enable RX channel to AUDIO. 0 = DISABLE 1 = ENABLE
29	0x0	LOOPBACK: Debug mode. The RX CIF is directly hooked up to the TX CIF. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
23:16	0x0	TX_THRESHOLD. Trigger threshold level, number of free slots - 1. DMA request is triggered when the number of free slots is > TX_THRESHOLD. For example, If a DMA request needs to be generated when 4 free slots are available, this field needs to be set to 3.
15:8	0x0	RX_THRESHOLD: Trigger threshold level, # of data DWords - 1. DMA request is triggered when the number of DWords in the FIFO is > RX_THRESHOLD. For example, if a DMA request needs to be generated when 4 DWords are available, this field needs to be set to 3.
6	0x0	TX_PACK_EN: Enable packed data. 0 = DISABLE 1 = ENABLE
5:4	0x0	TX_PACK: 0 = NOP 1 = RSVD 2 = PACK8_4 3 = PACK16
2	0x0	RX_PACK_EN: Enable packed data. 0 = DISABLE 1 = ENABLE
1:0	0x0	RX_PACK: 0 = NOP 1 = RSVD 2 = PACK8_4 3 = PACK16

20.10.3.2 APBIF2_CHANNELn_CLEAR_0

There are 6 APBIF2 Channel Clear Registers, one per channel (n = 4 through 9).

Offset: 0x4 + (n * 0x20) | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	TX_SOFT_RESET: 0 = DISABLE 1 = ENABLE
30	0x0	RX_SOFT_RESET: 0 = DISABLE 1 = ENABLE

20.10.3.3 APBIF2_CHANNELn_STATUS_0

There are 6 APBIF2 Channel Status Registers, one per channel (n = 4 through 9).

Offset: 0x8 + (n * 0x20) | Read/Write: RO | Reset: 0xXXXX000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	TX_FREE_COUNT: Number of free slots in the TX FIFO.
23:16	X	RX_FREE_COUNT: Number of free slots in the RX FIFO.
1	X	TX_TRIG: TX FIFO threshold. 0 = DISABLE 1 = ENABLE
0	X	RX_TRIG: RX FIFO threshold. 0 = DISABLE 1 = ENABLE

20.10.3.4 APBIF2_CHANNELn_TXFIFO_0

There are 6 APBIF2 Channel TX FIFO Registers, one per channel (n = 4 through 9).

Offset: 0xc + (n * 0x20) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

20.10.3.5 APBIF2_CHANNELn_RXFIFO_0

There are 6 APBIF2 Channel RX FIFO Registers, one per channel (n = 4 through 9).

Offset: 0x10 + (n * 0x20) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

20.10.3.6 APBIF2_AUDIOCIF_TXn_CTRL_0

There are 6 APBIF2 Audio CIF TX Control Registers, one per channel (n = 4 through 9).

When packed mode is enabled for a transmit channel, data is unpacked based on the APBIF2 TX channel CLIENT_BITS field.

- CLIENT_BITS = 0: Reserved. Do not use because 4 bit mode is not supported by the audio cluster.
- CLIENT_BITS = 1: (8 bit mode) : Four 8-bit samples are packed into one 32 bit word.
- CLIENT_BITS = 2: (12 bit mode): Two 12-bit samples are packed into one 32 bit word.
- CLIENT_BITS = 3: (16 bit mode): Two 16-bit samples are packed into one 32 bit word.
- CLIENT_BITS > 3: Packing is not possible.

Offset: 0x14 + (n * 0x20) | Read/Write: R/W | Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10

Bit	R/W	Reset	Description
			10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

20.10.3.7 APBIF2_AUDIOCIF_RXn_CTRL_0

There are 6 APBIF2 Audio CIF RX Control Registers, one per channel (n = 4 through 9).

Offset: 0x18 + (n * 0x20) | Read/Write: R/W | Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	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 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF

Bit	R/W	Reset	Description
			1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

20.10.3.8 APBIF2_AFC_LIVE_TX_FIFO_EMPTY_STATUS_0

Offset: 0xc0 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
5	X	AFC5_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
4	X	AFC4_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
3	X	AFC3_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
2	X	AFC2_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
1	X	AFC1_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	AFC0_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

20.10.3.9 APBIF2_AFC_LIVE_TX_FIFO_FULL_STATUS_0

Offset: 0xc4 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
5	X	AFC5_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
4	X	AFC4_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
3	X	AFC3_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
2	X	AFC2_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
1	X	AFC1_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
0	X	AFC0_TX_FIFO_FULL: 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE

20.10.3.10 APBIF2_AFC_LIVE_RX_FIFO_EMPTY_STATUS_0

Offset: 0xc8 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
5	X	AFC5_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
4	X	AFC4_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
3	X	AFC3_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
2	X	AFC2_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
1	X	AFC1_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	AFC0_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

20.10.3.11 APBIF2_AFC_LIVE_RX_FIFO_FULL_STATUS_0

Offset: 0xcc | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
5	X	AFC5_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
4	X	AFC4_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
3	X	AFC3_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
2	X	AFC2_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
1	X	AFC1_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
0	X	AFC0_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE

20.10.3.12 APBIF2_AFC_LIVE_TX_ENABLED_STATUS_0

Offset: 0xd0 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
5	X	AFC5_TX_ENABLED: 0 = DISABLE 1 = ENABLE
4	X	AFC4_TX_ENABLED: 0 = DISABLE 1 = ENABLE
3	X	AFC3_TX_ENABLED: 0 = DISABLE 1 = ENABLE
2	X	AFC2_TX_ENABLED: 0 = DISABLE 1 = ENABLE
1	X	AFC1_TX_ENABLED: 0 = DISABLE 1 = ENABLE
0	X	AFC0_TX_ENABLED: 0 = DISABLE 1 = ENABLE

20.10.3.13 APBIF2_AFC_TX_DONE_INT_MASK_0

Offset: 0xd4 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5	0x1	AFC5_TX_DONE: 0 = DISABLE 1 = ENABLE
4	0x1	AFC4_TX_DONE: 0 = DISABLE 1 = ENABLE
3	0x1	AFC3_TX_DONE: 0 = DISABLE 1 = ENABLE
2	0x1	AFC2_TX_DONE: 0 = DISABLE 1 = ENABLE
1	0x1	AFC1_TX_DONE: 0 = DISABLE 1 = ENABLE
0	0x1	AFC0_TX_DONE: 0 = DISABLE 1 = ENABLE

20.10.3.14 APBIF2_AFC_FLOW_CTL_INT_MASK_0

Offset: 0xd8 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5	0x1	AFC5_FLOW_CTL: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	0x1	AFC4_FLOW_CTL: 0 = DISABLE 1 = ENABLE
3	0x1	AFC3_FLOW_CTL: 0 = DISABLE 1 = ENABLE
2	0x1	AFC2_FLOW_CTL: 0 = DISABLE 1 = ENABLE
1	0x1	AFC1_FLOW_CTL: 0 = DISABLE 1 = ENABLE
0	0x1	AFC0_FLOW_CTL: 0 = DISABLE 1 = ENABLE

20.10.3.15 APBIF2_AFC_TX_DONE_INT_STATUS_0

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

Bit	Reset	Description
5	0x0	AFC5_TX_DONE: 0 = DISABLE 1 = ENABLE
4	0x0	AFC4_TX_DONE: 0 = DISABLE 1 = ENABLE
3	0x0	AFC3_TX_DONE: 0 = DISABLE 1 = ENABLE
2	0x0	AFC2_TX_DONE: 0 = DISABLE 1 = ENABLE
1	0x0	AFC1_TX_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	AFC0_TX_DONE: 0 = DISABLE 1 = ENABLE

20.10.3.16 APBIF2_AFC_FLOW_CTL_INT_STATUS_0

Offset: 0xe0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5	0x0	AFC5_FLOW_CTL: 0 = DISABLE 1 = ENABLE
4	0x0	AFC4_FLOW_CTL: 0 = DISABLE 1 = ENABLE
3	0x0	AFC3_FLOW_CTL: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	AFC2_FLOW_CTL: 0 = DISABLE 1 = ENABLE
1	0x0	AFC1_FLOW_CTL: 0 = DISABLE 1 = ENABLE
0	0x0	AFC0_FLOW_CTL: 0 = DISABLE 1 = ENABLE

20.10.3.17 APBIF2_AFC_TX_DONE_INT_SOURCE_0

Offset: 0xe4 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
5	X	AFC5_TX_DONE: 0 = DISABLE 1 = ENABLE
4	X	AFC4_TX_DONE: 0 = DISABLE 1 = ENABLE
3	X	AFC3_TX_DONE: 0 = DISABLE 1 = ENABLE
2	X	AFC2_TX_DONE: 0 = DISABLE 1 = ENABLE
1	X	AFC1_TX_DONE: 0 = DISABLE 1 = ENABLE
0	X	AFC0_TX_DONE: 0 = DISABLE 1 = ENABLE

20.10.3.18 APBIF2_AFC_FLOW_CTL_INT_SOURCE_0

Offset: 0xe8 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
5	X	AFC5_FLOW_CTL: 0 = DISABLE 1 = ENABLE
4	X	AFC4_FLOW_CTL: 0 = DISABLE 1 = ENABLE
3	X	AFC3_FLOW_CTL: 0 = DISABLE 1 = ENABLE
2	X	AFC2_FLOW_CTL: 0 = DISABLE 1 = ENABLE
1	X	AFC1_FLOW_CTL: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	X	AFC0_FLOW_CTL: 0 = DISABLE 1 = ENABLE

20.10.3.19 APBIF2_AFC_TX_DONE_INT_SET_0

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

Bit	Reset	Description
5	0x0	AFC5_TX_DONE: 0 = DISABLE 1 = ENABLE
4	0x0	AFC4_TX_DONE: 0 = DISABLE 1 = ENABLE
3	0x0	AFC3_TX_DONE: 0 = DISABLE 1 = ENABLE
2	0x0	AFC2_TX_DONE: 0 = DISABLE 1 = ENABLE
1	0x0	AFC1_TX_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	AFC0_TX_DONE: 0 = DISABLE 1 = ENABLE

20.10.3.20 APBIF2_AFC_FLOW_CTL_INT_SET_0

Offset: 0xf0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5	0x0	AFC5_FLOW_CTL: 0 = DISABLE 1 = ENABLE
4	0x0	AFC4_FLOW_CTL: 0 = DISABLE 1 = ENABLE
3	0x0	AFC3_FLOW_CTL: 0 = DISABLE 1 = ENABLE
2	0x0	AFC2_FLOW_CTL: 0 = DISABLE 1 = ENABLE
1	0x0	AFC1_FLOW_CTL: 0 = DISABLE 1 = ENABLE
0	0x0	AFC0_FLOW_CTL: 0 = DISABLE 1 = ENABLE

20.10.3.21 APBIF2_MISC_LIVE_STATUS0_0

Offset: 0xf4 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
5	X	AFC5_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
4	X	AFC4_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
3	X	AFC3_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
2	X	AFC2_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
1	X	AFC1_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
0	X	AFC0_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE

20.10.4 DAM Registers

20.10.4.1 DAM_CTRL_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31	DISABLE	SOFT_RESET: This bit is Auto Cleared. It resets the DAM logic including CIFs and only clears the DAM Enable & Channel Enables configuration bits. Software should wait until this bit is cleared. 0 = DISABLE 1 = ENABLE
7:4	X	FSOUT: Output DATA Sample Rate. 0 = FS8 1 = FS16 2 = FS44 3 = FS48 4 = FS11 5 = FS22 6 = FS24 7 = FS32 8 = FS88 9 = FS96 10 = FS176 11 = FS192
3	DISABLE	STEREO_MIXING_EN: Enables stereo mixing. IMPORTANT: This requires DAM to be in bypass mode! Therefore FSOUT and FSIN must be equal for this to work. 0 = DISABLE 1 = ENABLE
2	DISABLE	STEREO_SRC_EN: Enables stereo SRC. 0 = DISABLE 1 = ENABLE
1	DISABLE	CG_EN: Enables DAM second level clock gating 0 = DISABLE 1 = ENABLE
0	DISABLE	ENABLE: Enables the DAM 0 = DISABLE 1 = ENABLE

20.10.4.2 DAM_AUDIOCIF_OUT_CTRL_0

DAM Output

The DAM output is fixed at 32 bits. CIF should convert the output to the desired format

CLIENT_BITS : BIT32

Offset: 0xc | Read/Write: R/W | Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32

Bit	R/W	Reset	Description
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

20.10.4.3 DAM_CH0_CTRL_0

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00000xxx00000000xxx0)

Bit	Reset	Description
19:16	0x0	FILT_STAGES: Number of filter stages minus 1 (excluding interpolation/decimation stages). For example for an 8->48 conversion, there are two IIR filter stages. Therefore 0x1 should be programmed. 8->48 : 1 8->44.1 : 2 16->48 : 0 16->44.1 : 3 44.1->16 : 2 44.1->8 : 2 48->16 : 0 48->8 : 1 0 = ONE 1 = TWO 2 = THREE 3 = FOUR 4 = FIVE 5 = SIX 6 = SEVEN 7 = EIGHT
11:8	X	FSIN: Input sample rate 0 = FS8 1 = FS16 2 = FS44 3 = FS48 4 = FS11 5 = FS22 6 = FS24 7 = FS32 8 = FS88 9 = FS96 10 = FS176 11 = FS192
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 represents each channel, i.e., bit 1 -> CH 1. Bit 0 is meaningless for CH0. For CH0, 0b0000 is recommended since CH0 data should not wait for other channels.
0	DISABLE	ENABLE: 0 = DISABLE 1 = ENABLE

20.10.4.4 DAM_AUDIOCIF_CH0_CTRL_0

The DAM has 4 RX ports and 1 TX port.

Since LLF can handle only 16-bit mono data, CIF should be configured accordingly.

CLIENT_BITS : BIT16
CLIENT_CHANNELS : CH1
PACK : NOP

Offset: 0x1c | Read/Write: R/W | Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28

Bit	R/W	Reset	Description
			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

20.10.4.5 DAM_CH1_CTRL_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0001xxx0)

Bit	Reset	Description
7:4	0x1	DATA_SYNC: When a sample is ready in this channel, the bit says the mixer should wait for data in other channels. Each bit represents each channel, i.e., bit 0 -> CH 0. Bit 1 is meaningless for CH1. For CH1, 0b0001 is recommended since CH1 data should wait for CH0 data.
0	DISABLE	ENABLE: 0 = DISABLE 1 = ENABLE

20.10.4.6 DAM_AUDIOCIF_CH1_CTRL_0

Channel1 Input

Offset: 0x2c | Read/Write: R/W | Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11

Bit	R/W	Reset	Description
			11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO

Bit	R/W	Reset	Description
			1 = COPY

20.10.5 I2S Registers

The I^2S Controller is designed to transport streaming audio-data between the system memory and an audio-codec. The controller supports I^2S format, Left Justified Mode format, Right Justified Mode format, and DSP mode format, as defined in the Philips inter-IC-sound (I^2S) bus specification.

The controller also supports the PCM and telephony (network) mode of data-transfer. Pulse-Code-Modulation (PCM) is a standard method used to digitize audio (particularly voice) patterns for transmission over digital communication channels.

The Telephony (network) mode is used to transmit and receive data to/from an external mono codec in a slot-based scheme of time-division multiplexing.

The controller has one transmit and one receive interface. The controller can be configured to operate either as a master or a slave.

20.10.5.1 I2S_CTRL_0

The I2S control register is used to configure bit formats (I2S, LJM, RJM, DSP, PCM, NW, TDM), bit sizes (8, 16, 20, 24 and 32), FIFO formats, Master/slave selection, LR polarity, and error interrupt enables.

I2S Control Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0b00000xxxxxxx0xxxx000x000xx00x000)

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
19	0x0	PIPE_MACRO_EN: Enable pipe macro stage (1 stage in Tx direction and 1 in Rx direction). 0 = DISABLE 1 = ENABLE
14:12	0x0	FRAME_FORMAT: Frame format. 0: BASIC, LJM, RJM modes 1: DSP, PCM, NW, TDM modes 0 = LRCK_MODE 1 = FSYNC_MODE

Bit	Reset	Description
10	0x0	MASTER: Controller Master/Slave mode selection. 0: Do not drive LRCK and clock. 1: Drive LRCK and clock 0 = DISABLE 1 = ENABLE
9	0x0	LRCK_POLARITY: Left/Right Control Polarity. 0= Left channel when LRCK is low 1= Left channel when LRCK is high 0 = LOW 1 = HIGH
8	0x0	LPBK: Tx->Rx Loop Back Enable 0 = DISABLE 1 = ENABLE
5:4	0x0	BIT_CODE: sample compression/decompression 0: None 1: uLaw 2: aLaw 3: Reserved 0 = LINEAR 1 = uLAW 2 = ALAW 3 = RSVD
2:0	0x0	BIT_SIZE: Bit size. 0 = BIT_SIZE_RSVD 1 = BIT_SIZE_8 2 = BIT_SIZE_12 3 = BIT_SIZE_16 4 = BIT_SIZE_20 5 = BIT_SIZE_24 6 = BIT_SIZE_28 7 = BIT_SIZE_32

20.10.5.2 I2S_TIMING_0

The CHANNEL_BIT_CNT field of this register is used to program the number of bit-clocks per channel of LRCK (sampling rate) that the I^2S controller needs to send out in Master mode. This value is interpreted as follows:

- DSP mode: - Number of bit clocks (sclk) within (LEFT +RIGHT) channel width.
- Basic/LJM/RJM modes:- Number of bit clocks (sclk) within each individual channel width (LEFT or RIGHT).
- PCM/NW/TDM modes:- Number of bit clocks (sclk) per frame

The NON_SYM.EN can be used to program the I2S Controller to output a non-50:50 mark-space ratio on to the I2S bus. When the NON_SYM.EN Bit[12] is enabled, the Controller sends out exactly the programmed number of clock cycles on the left channel and one bit clock greater on the right channel. . This is used in Basic/LJM/RJM modes.

When NON_SYM.EN is enabled, the channel-bit-count should be programmed with the number of bit-clocks required in left-channel. This will ensure that the non-50:50 mark-space ratio is achieved with R_BCLK = L_BCLK + 1

The PCM/NW/TDM modes channel_bit_cnt can be calculated using the equation:

```
Channel_bit_cnt  = (frequency of bit_clk) / (2 * required sampling rate) -1;
```

If this calculation returns a fractional value, the non-symmetry feature of the controller should be used to attain the required sampling rate.

I2S Timing Register

Offset: 0x4 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxx0x00000011111)

Bit	Reset	Description
12	0x0	NON_SYM_EN: To enable non-symmetry mode. 0 = DISABLE 1 = ENABLE
10:0	0x1f	CHANNEL_BIT_CNT: I2S, LJM, RJM mode: Number of bit clocks in left or right channel. DSP mode: Number of bit clocks in LEFT+RIGHT channel TDM/NW/PCM mode: Number of bit clocks in the frame

20.10.5.3 I2S_OFFSET_0

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	RX_DATA_OFFSET: RX Data offset to fsync 0: No offset n: Data is offset by n bit clocks with respect to fsync
10:0	0x0	TX_DATA_OFFSET: TX Data offset to fsync 0: No offset n: Data is offset by n bit clocks with respect to fsync

20.10.5.4 I2S_CH_CTRL_0

Offset: 0xc | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxxxxxxxxx00x000x000x000)

Bit	Reset	Description
31:24	0x0	FSYNC_WIDTH: Fsync width in terms of bit clocks 0: Fsync width is 1 clock wide. 1: Fsync width is 2 clocks wide. n: Fsync width is n+1 clocks wide.
13:12	0x0	HIGHZ_CTRL: HighZ control 0: Always drive SdataOut. 1: Tristate SdataOut if a slot is not valid. 2: Tristate SdataOut after driving last bit data for half a bitclk cycle instead of driving it for a full bit clock 0 = NOHIGHZ 1 = HIGHZ 2 = HIGHZ_ON_HALF_BIT_CLK
10	0x0	RX_BIT_ORDER: Configure to appear MSB or LSB first on sdata out 0: MSB first 1: LSB first 0 = MSB_FIRST 1 = LSB_FIRST
9	0x0	TX_BIT_ORDER: Configure to appear MSB or LSB first on sdata out 0: MSB first 1: LSB first 0 = MSB_FIRST 1 = LSB_FIRST

Bit	Reset	Description
8	0x0	EDGE_CTRL: Edge on which Data is driven. 0: Drive data on the negative edge and sample data on the positive edge 1: Drive data on the positive edge and sample data on the negative edge 0 = POS_EDGE 1 = NEG_EDGE
6:4	0x0	RX_MASK_BITS: Used for PCM mode 0 = ZERO 1 = ONE 2 = TWO 3 = THREE 4 = FOUR 5 = FIVE 6 = SIX 7 = SEVEN
2:0	0x0	TX_MASK_BITS: Used for PCM mode. Set these mask bits to get the exact bit size in PCM mode. For example, to get 13 bits of data, BIT_SIZE should be set to 3 (BIT16) and MASK should be set to 3 (THREE). 0 = ZERO 1 = ONE 2 = TWO 3 = THREE 4 = FOUR 5 = FIVE 6 = SIX 7 = SEVEN

20.10.5.5 I2S_SLOT_CTRL_0

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	TOTAL_SLOTS: Number of slots per fsync 0: Frame has 1 slot 1: Frame has 2 slots n: Frame has n+1 slots

20.10.5.6 I2S_AUDIOCIF_I2STX_CTRL_0

An I2S module has two AUDIOCIF sub-modules. In default, the first AUDIO port is for RX and the second AUDIO port is for TX. However, the direction can be changed with register programming so that I2S can have either two RX or two RX AUDIO ports.

CIF RX Port for I2S TX Path

Offset: 0x14 │ Read/Write: R/W │ Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10

Bit	R/W	Reset	Description
			10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP

Bit	R/W	Reset	Description
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

20.10.5.7 I2S_AUDIOCIF_I2SRX_CTRL_0

CIF TX Port for I2S RX Path

Offset: 0x18 │ Read/Write: R/W │ Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24

Bit	R/W	Reset	Description
			6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

20.10.5.8 I2S_FLOWCTL_0

This register controls the flow control mechanism and is valid only with the TX mode. The flow control is only useful when the input stream to I^2S/SPDIF is already real-time spaced such as a live stream from Baseband through another I2S_RX. If the input stream is from DMA directly, the flow control should be disabled.

FLOWCTRL

Offset: 0x1c │ Read/Write: R/W │ Reset: 0x800XXXXX (0b10xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	QUAD	FILTER: Use linear or quadratic filter. Tegra K1 devices support quadratic filter only. 0 = LINEAR 1 = QUAD
30	DISABLE	DUMMY_WR_EN: Insert a dummy frame when enabled 0 = DISABLE 1 = ENABLE
17:16	X	START_THRESHOLD_MSB: Flow control should wait for FIFO filled higher 2 bits for threshold value
15:14	X	HIGH_THRESHOLD_MSB: high-threshold for HIGH state higher 2 bits for threshold value
13:12	X	LOW_THRESHOLD_MSB: low-threshold for LOW state higher 2 bits for threshold value
11:8	X	START_THRESHOLD: Flow control should wait for FIFO filled N-1, 0 means actual threshold = 1
7:4	X	HIGH_THRESHOLD: High-threshold for HIGH state N-1, 0 means actual threshold = 1
3:0	X	LOW_THRESHOLD: Low-threshold for LOW state N-1, 0 means actual threshold = 1

20.10.5.9 I2S_TX_STEP_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x00008000 (0bxxxxxxxxxxxxxxxx1000000000000000)

Bit	Reset	Description
15:0	0x8000	STEP_SIZE: This is a compensation value for the clock difference between input and output. To catch up the clock difference quickly, this value is supposed to be bigger than clock difference in terms of STEP_SIZE; however, to avoid sound quality degradation, it should not be much greater than the clock difference. For example, the input frame sampling rate is 44.095 kHz and the output frame sampling rate is 44.100 kHz, the clock error rate is (44100 - 44095) / 44100 = 0.000113. The step size used in the linear interpolation is 0x8000 (32768) step_size = 0.000113 * 32768 = 3.7. A threshold of 4 is recommended. If the value is less than 3 in the above example, the adjusted step size will not be able to catch up. The flow control FIFO will eventually become empty, leading to either NULL or replication of the last sample.

20.10.5.10 I2S_FLOW_STATUS_0

When the flow controller is used, the step size should be properly chosen depending on the clock frequency difference. The step size should be small enough not to degrade sound quality, but big enough to converge the FIFO depth.

If the step size is not big enough, the FIFO depth will either increase or decrease, which will cause FIFO overflow/underflow eventually. The flow controller monitor and counter can be used to fine-tune the step size. When the flow controller is enabled and a proper step size is chosen, the FIFO depth should be in [threshold -1 : threshold + 1].

When the flow controller monitor is enabled and the FIFO depth is deviated from the range, either FLOW_OVERFLOW or FLOW_UNDERFLOW will be set. Also, the interrupt signal will be asserted, if MONITOR_INT_EN is enabled. MONITOR_CLR clears the FLOW_OVERFLOW/UNDERFLOW as well as the interrupt signal to APBIF.

When the flow controller counter is enabled, it records how many samples are processed and how many times the step size is applied to either lower or raise the time step. In case of stereo samples, it counts as 1.

- FLOW_NORMAL - the total number of samples (only count left channel samples).
- FLOW_OVER - the number of samples with adding STEP_SIZE to the time step.
- FLOW_UNDER - the number of samples with subtracting STEP_SIZE from the time step.

Flow Controller Monitor/Counter

Offset: 0x24 | Read/Write: R/W | Reset: 0x0000000X (0b00xxxxxxxxxxxxxxxxxxxxxxxxxx0xx00)

Bit	Reset	Description
31	NORMAL	FLOW_UNDERFLOW: FIFO depth is less than threshold - 1 0 = NORMAL 1 = UNDER
30	NORMAL	FLOW_OVERFLOW: FIFO depth is greater than threshold + 1 0 = NORMAL 1 = OVER
4	DISABLE	MONITOR_INT_EN: Enable monitor interrupt to APBIF 0 = DISABLE 1 = ENABLE
3	X	COUNTER_CLR: Clear counter
2	X	MONITOR_CLR: Clear monitor
1	DISABLE	COUNTER_EN: Enable counter 0 = DISABLE 1 = ENABLE
0	DISABLE	MONITOR_EN: Enable monitor 0 = DISABLE 1 = ENABLE

20.10.5.11 I2S_FLOW_TOTAL_0

Offset: 0x28 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNTER: the total number of left channel samples processed

20.10.5.12 I2S_FLOW_OVER_0

Offset: 0x2c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNTER: the number of +STEP_SIZEs

20.10.5.13 I2S_FLOW_UNDER_0

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

Bit	Reset	Description
31:0	X	COUNTER: the number of -STEP_SIZEs

20.10.5.14 I2S_SLOT_CTRL2_0

Offset: 0x64 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	RX_SLOT_ENABLES: Rx Slot enables. Used for TDM mode
15:0	0x0	TX_SLOT_ENABLES: Tx Slot enables. Used for TDM mode Only the enabled slots are used to transmit or receive data in the TDM mode.

20.10.6 SPDIF Registers

The S/PDIF consists of the following two major modules:

- The SPDIFOUT sub-module, which sends data to the "spdifout" port in IEC 60958-3 biphase-mark code format.
- The SPDIFIN sub-module, which retrieves data to the "spdifin" port in IEC 60958-3 biphase-mark code format.

20.10.6.1 SPDIF_CTRL_0

SPDIF Control Register

Note: Changing the state of TC_EN, TU_EN, LBK_EN, PACK, BIT_MODE while RX_EN and/or TX_EN and/or RX_BSY and/or TX_BSY is set can cause unexpected behavior and therefore shall not be attempted.

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0b000000xxxxxxxxxx00000xxx0xxxxxxx)

Bit	Reset	Description
31	0x0	FLOWCTL_EN: 1=Enable flow control 0 = DISABLE 1 = ENABLE
30	0x0	CAP_LC: 1=start capturing from left channel,0=start capturing from right channel. 0 = RIGHT_CH 1 = LEFT_CH
29	0x0	RX_EN: SPDIF receiver (RX): 1=enable, 0=disable. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
28	0x0	TX_EN: SPDIF transmitter (TX): 1=enable, 0=disable. 0 = DISABLE 1 = ENABLE
27	0x0	TC_EN: Transmit Channel status: 1=enable, 0=disable. 0 = DISABLE 1 = ENABLE
26	0x0	TU_EN: Transmit user Data: 0 = DISABLE 1 = ENABLE
15	0x0	LBK_EN: Loopback test mode: 1=Enable internal loopback, 0=Normal mode. 0 = DISABLE 1 = ENABLE
14	0x0	PACK: Pack data mode: 1=Packeted left/right channel data into a single word, 0=Single data (16 bits need to be padded to match the interface data bit size) 0 = DISABLE 1 = ENABLE
13:12	0x0	BIT_MODE: 00=16-bit data. 01=20-bit data. 10=24-bit data. 11=raw data. This value should match ACIF CLIENT_BITS. 0 = MODE16BIT 1 = MODE20BIT 2 = MODE24BIT 3 = MODERAW
11	0x0	CG_EN: 1=Enable second level clock gating 0 = DISABLE 1 = ENABLE
7	0x0	SOFT_RESET: This bit is Auto Cleared. Resets I2S logic including CIFs and flow control. Configuration registers are not reset by soft reset. 0 = DISABLE 1 = ENABLE

20.10.6.2 SPDIF_STROBE_CTRL_0

SPDIF Data Strobe Control Register

Offset: 0x4 │ Read/Write: R/W │ Reset: 0x00XX0000 (0bxxxxxxxxxxxxxxxxx0xx00000xx000000)

Bit	R/W	Reset	Description
23:16	RO	X	PERIOD: Indicates the approximate number of detected SPDIFIN clocks within a biphase period.
15	RW	0x0	STROBE: SPDIFIN Data Strobe Mode 1=Manual-locked strobe 0=Auto-locked strobe (default)
12:8	RW	0x0	DATA_STROBES: Manual data strobe time within the biphase clock period (in terms of the number of oversampling clocks)
5:0	RW	0x0	CLOCK_PERIOD: Manual SPDIFIN biphase clock period (in terms of the number of over-sampling or 'spdifin' clocks)

20.10.6.3 SPDIF_AUDIOCIF_TXDATA_CTRL_0

An SPDIF module has four AUDIOCIF interfaces for TX data, RX data, TX user data, and RX user data. CIF RX port for SPDIF TX DATA

Offset: 0x8 │ Read/Write: R/W │ Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32

Bit	R/W	Reset	Description
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

20.10.6.4 SPDIF_AUDIOCIF_RXDATA_CTRL_0

CIF TX port for SPDIF RX DATA

Offset: 0xc │ Read/Write: R/W │ Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11

Bit	R/W	Reset	Description
			11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

20.10.6.5 SPDIF_AUDIOCIF_TXUSER_CTRL_0

CIF RX Port for SPDIF USER Data

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4

Bit	R/W	Reset	Description
			4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

20.10.6.6 SPDIF_AUDIOCIF_RXUSER_CTRL_0

CIF TX Port for SPDIF USER Data

Offset: 0x14 │ Read/Write: R/W │ Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32

Bit	R/W	Reset	Description
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

20.10.6.7 SPDIF_CH_STA_RX_A_0

This 6-word receive channel data page buffer holds a block (192 frames) of channel status information. The order of receive is from LSB bit to MSB bit, and from CH_STA_RX_A to CH_STA_RX_F and back to CH_STA_RX_A.

Note: Only channel status bits from channel A (subframe 1) will be saved into this page buffer.

SPDIF Channel Status Rx Page Buffer Register

Offset: 0x18 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RX_C31_RX_C0: Channel status bits [31:0]; one bit per audio sample

20.10.6.8 SPDIF_CH_STA_RX_B_0

Offset: 0x1c │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RX_C63_RX_C32: Channel status bits [63:32]; one bit per audio sample

20.10.6.9 SPDIF_CH_STA_RX_C_0

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

Bit	Reset	Description
31:0	X	RX_C95_RX_C64: Channel status bits [95:64]; one bit per audio sample

20.10.6.10 SPDIF_CH_STA_RX_D_0

Offset: 0x24 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RX_C127_RX_C96: Channel status bits [127:96]; one bit per audio sample

20.10.6.11 SPDIF_CH_STA_RX_E_0

Offset: 0x28 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RX_C159_RX_C128: Channel status bits [159:128]; one bit per audio sample

20.10.6.12 SPDIF_CH_STA_RX_F_0

Offset: 0x2c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RX_C191_RX_C160: Channel status bits [191:160]; one bit per audio sample

20.10.6.13 SPDIF_CH_STA_TX_A_0

This 6-word transmit channel data page buffer holds a block (192 frames) of channel status information. The order of transmission is from LSB bit to MSB bit, and from CH_STA_TX_A to CH_STA_TX_F and back to CH_STA_TX_A.

Note: The channel status data from this page buffer will be used only when PACK=1, or when BIT_MODE=00/01/10.
Each channel status bit will be sent out to "both" subframes.

SPDIF Channel Status Tx Page Buffer Register

Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_C31_TX_C0: Channel status bits [31:0]; one bit per audio sample

20.10.6.14 SPDIF_CH_STA_TX_B_0

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

Bit	Reset	Description
31:0	0x0	TX_C63_TX_C32: Channel status bits [63:32]; one bit per audio sample

20.10.6.15 SPDIF_CH_STA_TX_C_0

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

Bit	Reset	Description
31:0	0x0	TX_C95_TX_C64: Channel status bits [95:64]; one bit per audio sample

20.10.6.16 SPDIF_CH_STA_TX_D_0

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

Bit	Reset	Description
31:0	0x0	TX_C127_TX_C96: Channel status bits [127:96]; one bit per audio sample

20.10.6.17 SPDIF_CH_STA_TX_E_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_C159_TX_C128: Channel status bits [159:128]; one bit per audio sample

20.10.6.18 SPDIF_CH_STA_TX_F_0

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

Bit	Reset	Description
31:0	0x0	TX_C191_TX_C160: Channel status bits [191:160]; one bit per audio sample

20.10.6.19 SPDIF_FLOWCTL_CTRL_0

This register controls the flow control mechanism and is valid only with the TX mode. The flow control is only useful when the input stream to I2S/SPDIF is already real-time spaced such as a live stream from Baseband through another I2S_RX. If the input stream is from DMA directly, the flow control should be disabled.

Offset: 0x70 | Read/Write: R/W | Reset: 0x80000XXX (0b10xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	QUAD	FILTER: to use linear or quadratic filter. Tegra K1 devices support quadratic filter only. 0 = LINEAR 1 = QUAD
30	DISABLE	DUMMY_WR_EN: Insert a dummy frame when enabled 0 = DISABLE 1 = ENABLE
11:8	X	START_THRESHOLD: Flow control should wait for FIFO filled N-1, 0 means actual threshold 1
7:4	X	HIGH_THRESHOLD: High-threshold for HIGH state N-1, 0 means actual threshold 1
3:0	X	LOW_THRESHOLD: Low-threshold for LOW state N-1, 0 means actual threshold 1

20.10.6.20 SPDIF_TX_STEP_0

Offset: 0x74 │ Read/Write: R/W │ Reset: 0x00008000 (0bxxxxxxxxxxxxxxxx1000000000000000)

Bit	Reset	Description
15:0	0x8000	STEP_SIZE: This is a compensation value for the clock difference between input and output. To catch up the clock difference quickly, this value is supposed to be bigger than clock difference in terms of STEP_SIZE, but to avoid the sound quality degradation it should not be much greater than the clock difference. For example, the input frame sampling rate is 44.095 kHz, the output frame sampling rate is 44.100 kHz, and the clock error rate is (44100 - 44095) / 44100 = 0.00011.3 The step size used in the linear interpolation is 0x8000 (32768) step_size = 0.000113 * 32768 = 3.7. A threshold of 4 is recommended. If the value is less than 3 in the above example, the adjusted step size will not be able to catch up. The flow control FIFO will eventually become empty, leading to either NULL or replication of the last sample.

20.10.6.21 SPDIF_FLOW_STATUS_0

When the flow controller is used, the step size should be properly chosen depending on the clock frequency difference. The step size should be small enough not to degrade sound quality, but big enough to converge the FIFO depth. If the step size is not big enough, the FIFO depth will either increase or decrease, which will cause FIFO overflow/underflow eventually.

The flow controller monitor and counter can be used to fine-tune the step size. When the flow controller is enabled and a proper step size is chosen, the FIFO depth should be in [threshold -1 : threshold + 1].

When the flow controller monitor is enabled and the FIFO depth is deviated from the range, either FLOW_OVERFLOW or FLOW_UNDERFLOW will be set. Also, the interrupt signal will be asserted, if MONITOR_INT_EN is enabled. MONITOR_CLR clears the FLOW_OVERFLOW/UNDERFLOW as well as the interrupt signal to APBIF.

When the flow controller counter is enabled, it records how many samples are processed and how many times the step size is applied to either lower or raise the time step. In case of stereo samples, it counts as 1.

- FLOW_NORMAL - the total number of samples (only count left channel samples).
- FLOW_OVER - the number of samples with adding STEP_SIZE to the time step.
- FLOW_UNDER - the number of samples with subtracting STEP_SIZE from the time step.

Flow Controller Monitor/Counter

Offset: 0x78 │ Read/Write: R/W │ Reset: 0x0000000X (0b00xxxxxxxxxxxxxxxxxxxxxxxxxx0xx00)

Bit	Reset	Description
31	NORMAL	FLOW_UNDERFLOW: FIFO depth is less than threshold - 1 0 = NORMAL 1 = UNDER
30	NORMAL	FLOW_OVERFLOW: FIFO depth is greater than threshold + 1 0 = NORMAL 1 = OVER
4	DISABLE	MONITOR_INT_EN: enable monitor interrupt to APBIF 0 = DISABLE 1 = ENABLE
3	X	COUNTER_CLR: clear counter
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

20.10.6.22 SPDIF_FLOW_TOTAL_0

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

Bit	Reset	Description
31:0	X	COUNTER: the total number of left channel samples processed

20.10.6.23 SPDIF_FLOW_OVER_0

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

Bit	Reset	Description
31:0	X	COUNTER: the number of +STEP_SIZEs

20.10.6.24 SPDIF_FLOW_UNDER_0

Offset: 0x84 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNTER: the number of -STEP_SIZEs

20.10.7 AMX Registers

20.10.7.1 AMX_CTRL_0

AMX Control Register

This register specifies the input stream parameters (the number of audio channels in the input stream and number of bits per sample).

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxx0000xxxxxxxx)

Bit	Reset	Description
31	0x0	SOFT_RESET: This bit is auto cleared. Resets AMX logic. 0 = DISABLE 1 = ENABLE
30	0x0	CG_EN: Second level clock gating enable. 0 = DISABLE 1 = ENABLE
11:10	0x0	MSTR_CH_NUM: Designated master channel. 0 = CH0 1 = CH1 2 = CH2 3 = CH3
9:8	0x0	CH_DEP: This field determines whether to wait for all enabled channels to have data before a transfer or to start sending when any one of the input channels has data. 0x0: Send output when ALL enabled channels have data. 0x1: Send output when any one of the enabled channels has data. 0x2: Send output when designated master stream has data. 0x3: Reserved 0 = WT_ON_ALL 1 = WT_ON_ANY 2 = WT_ON_MASTER 3 = RSVD

20.10.7.2 AMX_IN_CH_CTRL_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxx0000)

Bit	Reset	Description
11	0x0	CH3_FORCE_DISABLE: Channel 3 enable 0 = DISABLE 1 = ENABLE
10	0x0	CH2_FORCE_DISABLE: Channel 2 enable 0 = DISABLE 1 = ENABLE
9	0x0	CH1_FORCE_DISABLE: Channel 1 enable 0 = DISABLE 1 = ENABLE
8	0x0	CH0_FORCE_DISABLE: Channel 0 enable 0 = DISABLE 1 = ENABLE
3	0x0	CH3_EN: Channel 3 enable 0 = DISABLE 1 = ENABLE
2	0x0	CH2_EN: Channel 2 enable 0 = DISABLE 1 = ENABLE
1	0x0	CH1_EN: Channel 1 enable 0 = DISABLE 1 = ENABLE
0	0x0	CH0_EN: Channel 0 enable 0 = DISABLE 1 = ENABLE

20.10.7.3 AMX_OUT_BYTE_EN0_0

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

Bit	Reset	Description
31:0	0x0	BYTE_EN: Byte enables for bytes 0 to 31

20.10.7.4 AMX_OUT_BYTE_EN1_0

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

Bit	Reset	Description
31:0	0x0	BYTE_EN: Byte enables for bytes 32 to 63

20.10.7.5 AMX_AUDIORAMCTL_AMX_CTRL_0

Offset: 0x10 | Read/Write: R/W | Reset: 0xX0000000 (0bxxxxxxxxxxxxxxxxxx000xxxx00000000)

Bit	R/W	Reset	Description
31	RO	X	READ_BUSY: 0 = DONE 1 = BUSY
14	RW	0x0	RW: 0 = READ 1 = WRITE

Bit	R/W	Reset	Description
13	RW	0x0	RESET_HW_ADR: 0 = DONE 1 = BUSY
12	RW	0x0	HW_ADR_EN: 0 = DISABLE 1 = ENABLE
7:0	RW	0x0	RAM_ADR

20.10.7.6 AMX_AUDIORAMCTL_AMX_DATA_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA

20.10.7.7 AMX_AUDIOCIF_OUT_CTRL_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16

Bit	R/W	Reset	Description
			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

20.10.7.8 AMX_AUDIOCIF_CHn_CTRL_0

There are 4 AMX Audio CIF Channel Registers, one per channel (n = 0 through 3).

Offset: 0x1c + (n * 0x04) | Read/Write: R/W | Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15

Bit	R/W	Reset	Description
			15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

20.10.8 ADX Registers

20.10.8.1 ADX_CTRL_0

ADX Control Register

This register specifies the input stream parameters (the number of audio channels in the input stream and number of bits per sample).

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	SOFT_RESET: This bit is auto cleared. Resets ADX logic 0 = DISABLE 1 = ENABLE
30	0x0	CG_EN: Second-level clock gating enable. 0 = DISABLE 1 = ENABLE

20.10.8.2 ADX_OUT_CH_CTRL_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxx0000)

Bit	Reset	Description
11	0x0	CH3_FORCE_DISABLE: Ch 3 enable 0 = DISABLE 1 = ENABLE
10	0x0	CH2_FORCE_DISABLE: Ch 2 enable 0 = DISABLE 1 = ENABLE
9	0x0	CH1_FORCE_DISABLE: Ch 1 enable 0 = DISABLE 1 = ENABLE
8	0x0	CH0_FORCE_DISABLE: Ch 0 enable 0 = DISABLE 1 = ENABLE
3	0x0	CH3_EN: Ch3 enable 0 = DISABLE 1 = ENABLE
2	0x0	CH2_EN: Ch2 enable 0 = DISABLE 1 = ENABLE
1	0x0	CH1_EN: Ch1 enable 0 = DISABLE 1 = ENABLE
0	0x0	CH0_EN: Ch0 enable 0 = DISABLE 1 = ENABLE

20.10.8.3 ADX_IN_BYTE_EN0_0

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

Bit	Reset	Description
31:0	0x0	BYTE_EN: Byte enables for bytes 0 to 31

20.10.8.4 ADX_IN_BYTE_EN1_0

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

Bit	Reset	Description
31:0	0x0	BYTE_EN: Byte enables for bytes 32 to 63

20.10.8.5 ADX_AUDIORAMCTL_ADX_CTRL_0

Offset: 0x10 | Read/Write: R/W | Reset: 0xX0000000 (0bxxxxxxxxxxxxxxxxxx000xxxx00000000)

Bit	R/W	Reset	Description
31	RO	X	READ_BUSY: 0 = DONE 1 = BUSY
14	RW	0x0	RW: 0 = READ 1 = WRITE
13	RW	0x0	RESET_HW_ADR: 0 = DONE 1 = BUSY
12	RW	0x0	HW_ADR_EN: 0 = DISABLE 1 = ENABLE
7:0	RW	0x0	RAM_ADR

20.10.8.6 ADX_AUDIORAMCTL_ADX_DATA_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA

20.10.8.7 ADX_AUDIOCIF_IN_CTRL_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3

Bit	R/W	Reset	Description
			3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

20.10.8.8 ADX_AUDIOCIF_CHn_CTRL_0

There are 4 ADX Audio CIF Channel Registers, one per channel (n = 0 through 3).

Offset: 0x1c + (n * 0x04) | Read/Write: R/W | Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AUDIO_CHANNELS:

Bit	R/W	Reset	Description
			0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF

Bit	R/W	Reset	Description
			1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

21.0 DISPLAY CONTROLLER

The Tegra® K1 architecture has two entirely independent display controllers. They can support two independent display devices, typically a local display panel and an external HDMI TV. Other configurations are possible such as two local panels. Each display controller can run at a different clock rate and drive a different resolution panel.

The Tegra K1 display controllers are significantly more efficient in the way they fetch from memory compared with prior Tegra devices. They use line buffers to store scan lines required for re-use, to avoid redundant fetches when scaling or using a tiled surface format.

21.1 Features

Key features include:

- Additional serial output resource (SOR) to support LVDS, eDP concurrent with HDMI or MIPI-DSI
- Three full function display windows, A, B and C, supporting scaling and blending
- Two additional windows (collectively known as “Simple Windows”):
  - Window D: Fourth general-purpose window
  - Window T: TrustZone Secure display
- Wider horizontal filter coefficients
- Maximum panel resolution:
  - Internal panel
    - LVDS – 1920 x 1200 @ 60 Hz (2D – portrait/landscape)
    - DSI

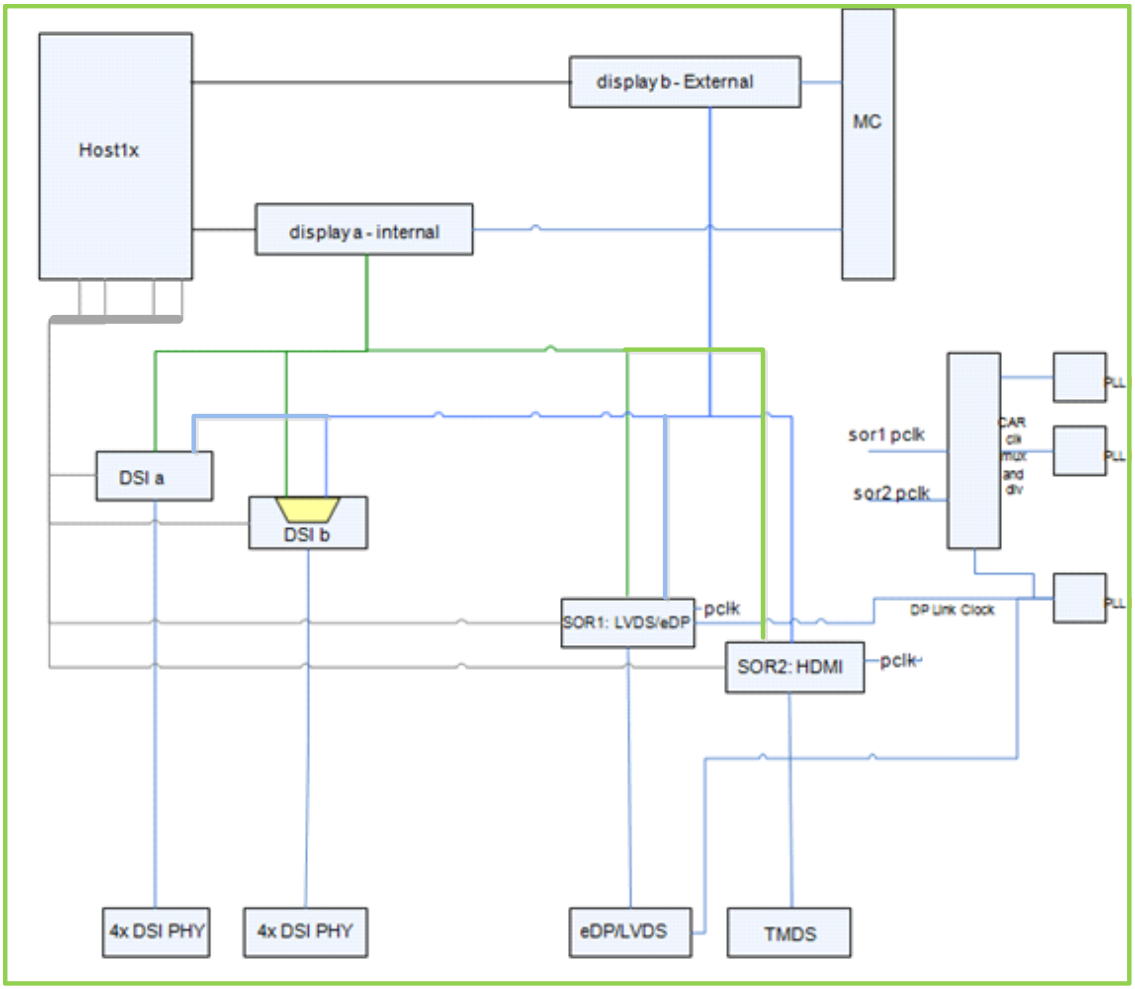
      Dual link 3200 x 2000 @ 60Hz (2D – portrait/landscape)

      Single link 2560 x 1440 @ 60Hz (2D – portrait/landscape)
    - eDP

      3840 x 2160 @ 60 Hz (2D – portrait/landscape)
  - External panel
    - HDMI 4096 x 2160 @ 30 Hz (2D – landscape)
    - HDMI 1920 x 1080 @ 60 Hz (3D – frame packed mode)
- Use of line buffering to avoid refetching data from memory on subsequent lines
- Line buffer free dithering: temporal dithering
- HDMI 1080i and HDMI YUV output support
- Efficient fetch for scaling (select windows)
- Improved handling of display buffer underflow (select windows)
- Supports high-quality scaling/filtering (select windows)
- Supports up to QHD resolution
- Support for Image Rotation in hardware
- Supports semi-planar YUV
- Extending the cursor size to 256x256 with full color and alpha-blending
- Display Color Management unit (based on a LUT and CSC in the output path)
- Improved layered blending among all windows

- Independent cursor update/activate control

Figure 52: Display Controller Top-Level Block Diagram



21.1.1 Output Capabilities

- 1 or 2 internal panels
  - MIPI-DSI on two 4-lane DSI channels, operating either independently for one or two panels, or ganged together as 2x4 for a single panel
    - Maximum panel size of 3200 x 2000 @ 60 Hz using 2x4-lane ganged DSI channels
    - Independent resolution and pixel clock when not ganged
  - LVDS supports 18/24 bpp, 3/4 data lanes, 1 clock lane, 1920 x 1200 @ 60 Hz
  - eDP supports 18/24 bpp, 1/2/4 data lanes, RBR/HBR/HBR2 up to 3840x2160 @ 60Hz
- 1 external display
  - On HDMI. Maximum 1920x1080p @ 60 Hz (3D) or 4096x2160 @ 30 Hz (2D)
  - With HDCP and multi-channel audio

21.1.2 Color Management Unit (CMU)

- Gamma (tone curve) conversion

- Color gamut conversion
- Allow conversion of sRGB content for a non-sRGB panel, for predictable colors (color matching)
- Enhanced PRISM 3 display for these panels, as the PRISM 3 algorithm assumes sRGB gamma
- Enable TCON and panels with non sRGB response
- Allows color calibrated displays, and enables end-to-end Camera / ISP -> Display color quality

21.1.3 Windows A/B/C

- Data Formats include (see the “Surface/Color Formats” subsection for a complete list):
  - B4G4R4A4
  - B5G5R5A, AB5G5R5, B5G6R5
  - B8G8R8A8, R8G8B8A8
  - YCbCr420P , YCbCr422P, YCbCr422RP (planar format)
  - YCbCr422 (packed format)
  - Palletized format limited to 8-bit mode.
- Three 256x8-bit RAMs (color palette) used for gamma correction.
- Horizontal and Vertical Flip
- Horizontal and Vertical Scaling
  - Horizontal scale-down by up to 4x, subject to clock speed limitations
  - Vertical scale-down by up to 4x for 2 bytes per pixel or 2x for 4 bytes per pixel, again subject to clock speed limitations
  - Scale-up by up to 4000x (from 1 pixel to 4000 pixels)
- Filtering (Windows A/B/C)
  - 6-tap, 16-phase programmable H filter
  - 2-tap, 16-phase programmable V filter
- YCbCr to RGB color space converter (Windows A/B/C)
- Digital Vibrance (8-level)

The table below describes a breakdown of features for a Tegra K1 Display on a window basis. Note that rotation is supported up to 2560x1600. Also note that system bandwidth, power, and other constraints might limit what combinations of resolution and features are achievable.

Table 74: Window Feature Summary

Head	Maximum Resolution	Window	Scaling	Planar Rotation	Packed Rotation
A	3840x2160 @ 60 Hz	A	Y	Y	Y
		B	Y	N	Y
		C	Y	N	Y
		D	N	N	N
		T	N	N	N
B	4096x2160 @ 30 Hz	A	Y	N	N
		B	Y	N	N
		C	Y	N	N

21.1.4 Windows D/T (Simple Windows)

Windows D and T are similar, generic windows with limited features for supporting status bars, soft buttons, and similar use cases. Window T has additional TrustZone semantics and limitations.

Windows D/T, collectively known as “simple windows”, support a subset of the registers used by the full-featured windows A/B/C.

The following table summarizes the differences among the two windows.

Table 75: Windows D/T Summary

Feature	Window D	Window T
32/16 bpp RGBA	Y	Y
Syncpt	Y	N
STATE_CONTROL shadow	Y	N
TrustZone Secure mode	N	Y
Indirect access (DISPLAY_WINDOW_HEADER)	Y	N
Class (Host1x command) access	Y	N [secure mode]

21.1.4.1 Feature Limitations

Windows D/T have similar window datapath features. Unlike window T, window D has syncpt increment and state control similar to windows A/B/C. Limitations of windows D/T relative to A/B/C include:

- Color depth is limited to a subset of RGB/RGBA single-plane formats. YCbCr/YUV/palette formats are not supported. The following formats are supported:
  - T_R8G8B8A8 – Tegra 4-style RGBA cursor uses
  - T_A8R8G8B8 – Tegra 3/Tegra 4 window A/B/C format
  - T_A8B8G8R8 – Tegra 3/Tegra 4 window A/B/C format
  - T_R5G6B5 – Best quality 16bpp format
  - T_A4R4G4B4 – 16bpp format with 4-bit alpha
  - T_A1R5G5B5 – 16bpp with 1-bit alpha format, better quality
- No scaling or filtering (DDA_INCREMENT, H/V_FILTER_ENABLE)
- Blending is limited to per-pixel alpha (ALPHA_WEIGHT and PREMULT_WEIGHT) and fixed alpha (FIX_WEIGHT). No color key or dependent weight (i.e., no dependence on non-secure pixels)
- The following features are not supported:
  - Block linear
  - H_DIRECTION, V_DIRECTION
  - Rotation (SCAN_COLUMN)
  - DV (digital vibrance)
  - BYTE_SWAP
  - H/V_INITIAL_DDA
  - DDA_INCREMENT (no scaling)

Window T has some additional feature limitations:

- No syncpt increment
- No indirect access using DISPLAY_WINDOW_HEADER – only window direct space is supported
- No class writes in secure mode. Class writes are always non-secure and therefore cannot be used in secure mode.
- Register activation and shadow control do not use STATE_CONTROL/STATE_ACCESS/REG_ACT_CONTROL. Separate registers are used.

21.1.4.2 Blending

Window D/T supports a subset of window blending registers. Because color key is not supported, only MATCH case exists.

21.1.4.3 Window T TrustZone Support

The display controller supports the ability to display TrustZone secured content in window T. Registers and memory interface in window T are separate from the other windows.

The display controller can switch dynamically between Secure and non-Secure modes. In Secure mode, window T is composited on top of 4 non-secure windows, and the cursor is disabled. The Non-secure OS can continue to run as normal, but its windows maybe obscured. All other semantics for the non-Secure OS are unchanged, including register reads, sync points, and interrupts.

TZ_SECURE is not intended for digital media DRM; the Video Protection Region is intended for that. The TZ_SECURE window T also does not have sufficient features for video use cases. TZ and VPR are orthogonal. TZ_SECURE does not require a VPR region, and VPR region access does not require TZ_SECURE.

Only displayA (internal display) has window T. But, both display A and displayB (external display) have TZ Secure host interfaces and implement the SECURE_CONTROL register. This allows the Secure OS to control the displayB output to DSI.

The display controller allows window T to use an alternate MC address-space ID (ASID) so that secure and non-secure address translation and attributes can be different. Window T is hardcoded to a different SWID (via the swid signal) from all other windows. Software can either program the two SWIDs to different ASIDs or to the same ASID.

The display controller allows disabling CRC in Secure mode, so that non-secure software cannot gain information about trusted display by examining the CRC.

Secure Mode Limitations

The TrustZone Secure OS mode has these limitations:

- No hardware cursor. The cursor could be used to spoof secure window.
- No use of display syncpt increment, interrupt or raise. The single interrupt line is shared by the whole head (all windows).
- DISPLAY_WINDOW_HEADER should not be used by TZ. TZ must use “window direct” address range only.
- Only direct Host1x writes can support TZ_SECURE accesses; indirect and CDMA/class writes might not.
- The Host1x channel should not be shared with non-TZ OS. If unavoidable, only Host1x direct read/write should be done. In particular, non-atomic Host1x sequences, such as indirect register writes, should be avoided.
- STATE_CONTROL, STATE_ACCESS, and REG_ACT_CONTROL must not be used by TZ. SECURE_STATE_CONTROL and WIN_T_STATE_ACCESS should be used instead.
- Display CRC includes the Secure Window which could confuse the non-Secure OS, if the latter uses CRC. This applies to functions like NvDPS.
- Secure window affects PRISM 3 and thus backlight brightness. It might interact poorly with SD_WINDOW* region, for example.

- Bandwidth used by window T must be accounted for in DVFS, IMP, etc.

21.1.4.4 DVFS Support

Windows D/T are also responsible for generating DVFS readiness signals similar to those generated by MEMFETCH windows. The requirements for generating ready_for_latency_event for windows D/T DVFS are the same as for full featured windows.

21.2 Block Diagrams

Figure 53: Display Controller Front End Block Diagram

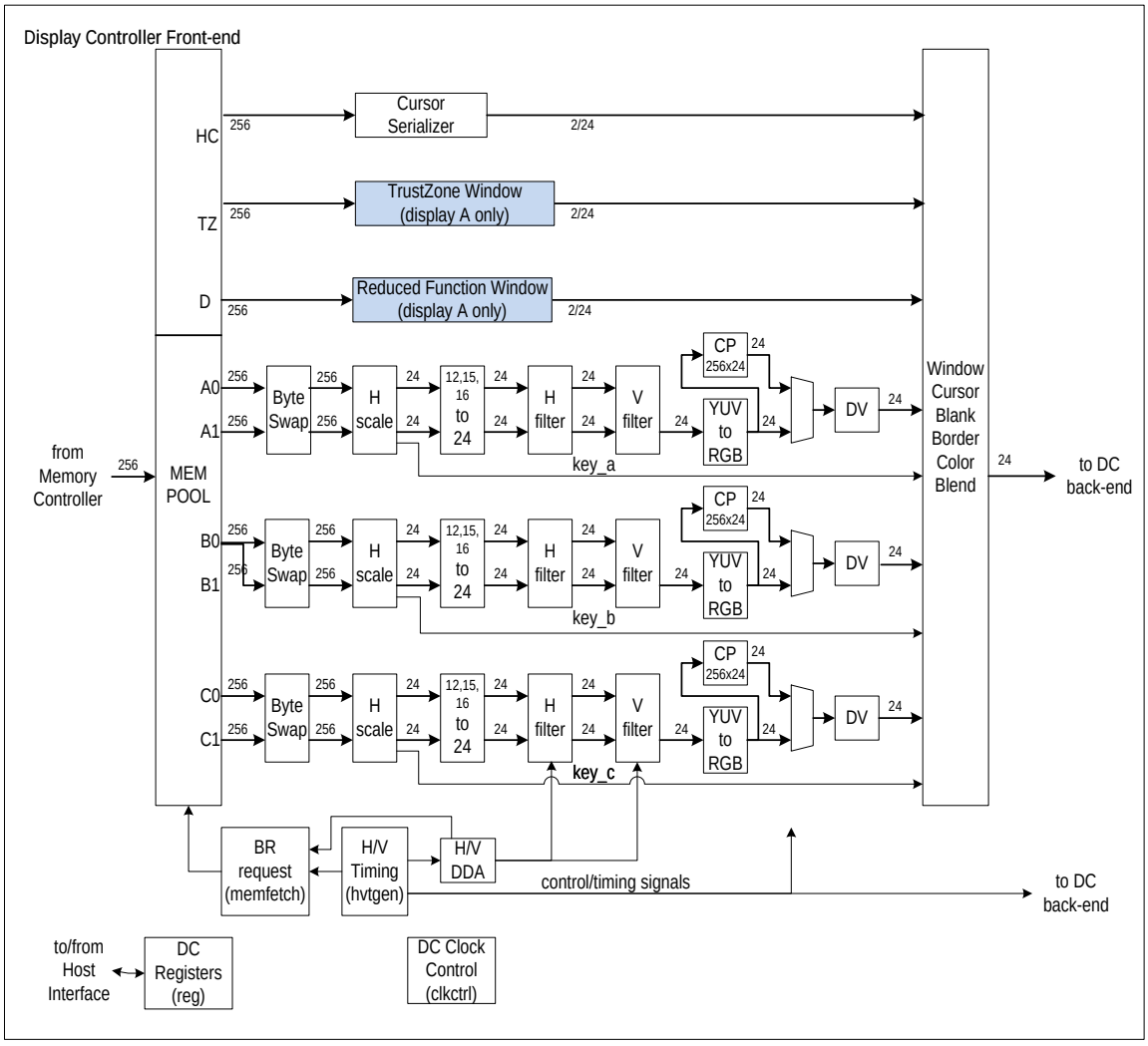
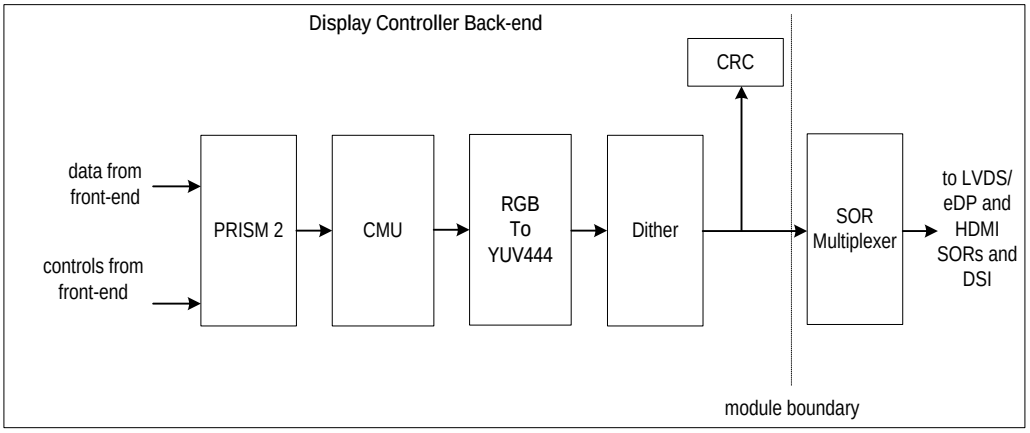


Figure 54: Display Controller Back End Block Diagram



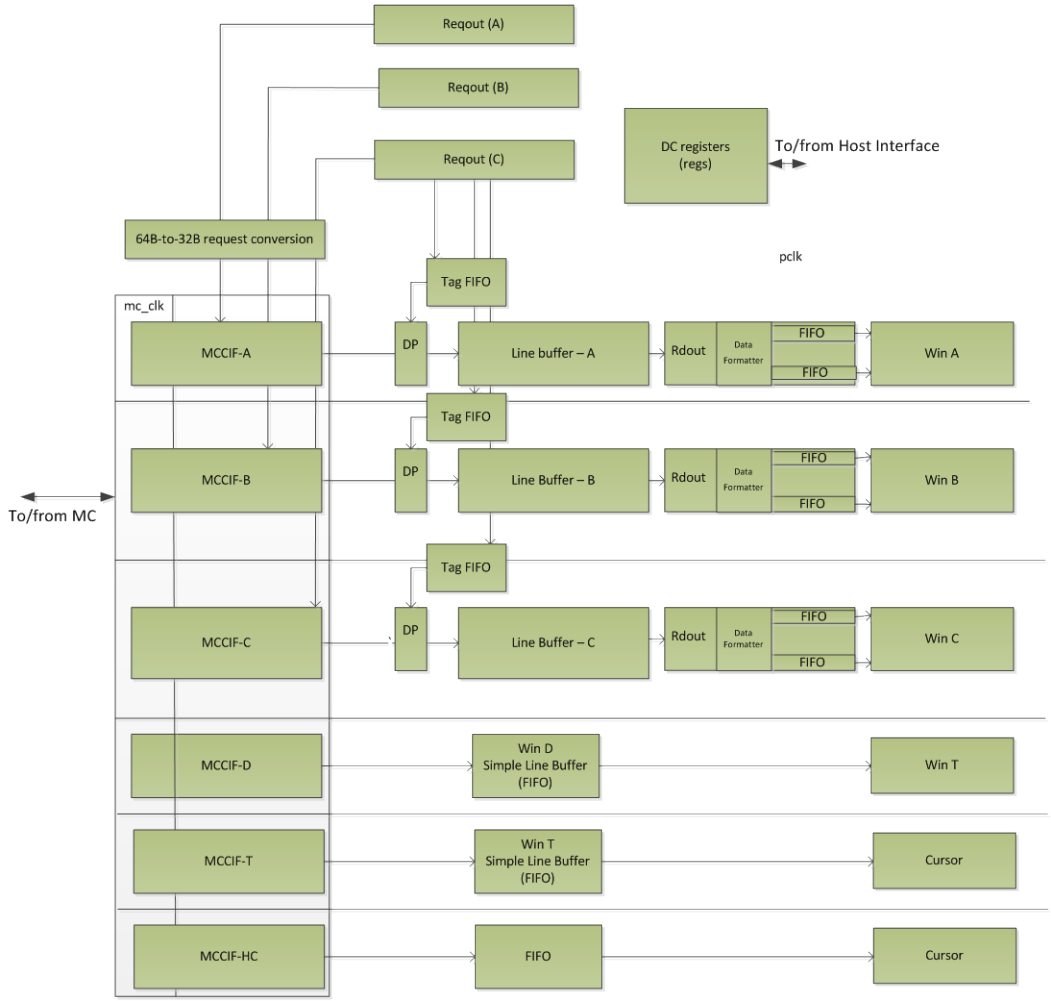
21.3 Display Controller Description

21.3.1 MEMFETCH

Figure 55 shows the top-level block diagram of the display memory interface for Tegra K1 devices. The diagram shows the memory fetch engines associated with the three full-featured windows and two simple windows in the display controller. Each window memory fetch engine has a line buffer and MCCIF. In addition to the windows, display also uses a memfetch for cursor.

Requests are issued to the memory and data is returned via MCCIF to the window line buffer. Data from the line buffer is subsequently read out, unpacked and passed to the window scaler and filter units. Windows D/T and the cursor fetch engines do not contain line buffers.

Figure 55: Memfetch Block Diagram



21.3.1.1 Request Engine

The request engine is responsible for fetching the window image from memory. Logic to translate the X, Y coordinates to a linear memory address is used to make requests to memory.

A pulse signal is sent to the request engine from the display timing generator at the beginning of the vertical blanking period to alert the request engine to start requesting the image data. Tagged requests are sent to the MCCIF. The thread ID specifying the line location of the return data is passed to the tag FIFO between the request and response engines. The thread ID is read by the data packer block when the return data is received and the data is steered to the appropriate line(s) in the line buffer.

The request engine must handle the following surface formats:

- Pitched linear surface
- Block linear surface

  16B x 2 lines – subset of 32B x 2 lines (rotation disabled)

  16B x 2 lines – subset of 32B x 4 lines (rotation enabled)

Requests are only stalled by:

- Backpressure from MCCIF
  - Occurs when MCCIF reorder buffer allocation is full

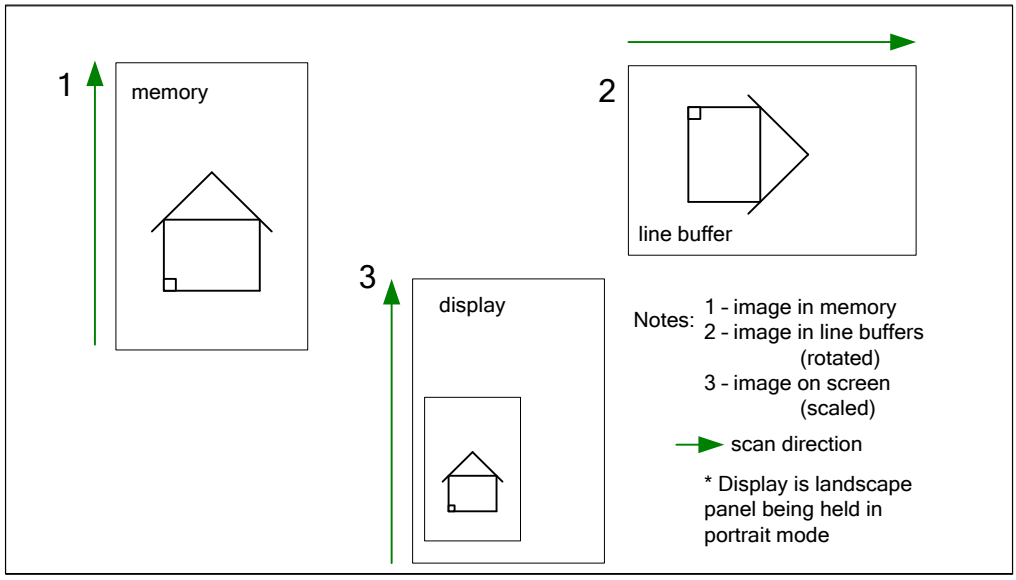
- MCCIF reorder buffer size is 6KB

- Tag FIFO full
  - Each tag represents 64 byte line buffer entry
  - Current tag FIFO depth sized at 119 entries
  - Allows for slightly over 7KB requests
  - Tag FIFO depth should be slightly greater than possible MCCIF outstanding requests

Rotation Support

The display supports 0°, 90°,180°, or 270° rotation or image flip. To support rotation, the memfetch unit rotates the data at the input to the line buffers. That is, memfetch walks the Y axis and stores these columns in memory as rows to the display. The figure below illustrates the image orientation and shape at different stages in the pipeline for rotation.

Figure 56: Example Rotation Use-Case



Memfetch only supports rotation in block linear mode. This is due to the amount of horizontal pixel data read in pitched linear versus block linear format. For pitched linear, 32B of horizontal data is fetched compared to block linear format where only 16B of horizontal data is fetched using the 16B x 4 line block linear mode. Since rotation switches rows and columns, the 4 horizontal pixels become 4 rows in the line buffer. Therefore the minimum number of lines when rotation is enabled is 4. The number of lines grows to 5 when scaling is enabled.

The request for 16Bx4 format must be communicated to the MCCIF via addressing bits 6 and 7. Memfetch will increment the Y instead of X coordinate for subsequent accesses and after reaching maximum Y will reset the coordinates to (X+16B, 0) for the next fetch of columns from memory. Selection of 32Bx2 or 16Bx4 is based on B_SCAN_COLUMN used to indicate rotation

The B_SCAN_COLUMN bit indicates if the image needs to be scanned out row by row or column by column. When this bit is enabled, the display needs to scan out lines column by column. This will provide the required 90°/270° rotated images. To achieve 90° or 270° rotation, software must use a combination of H_DIR, V_DIR, and SCAN_COLUMN as shown in the table below. It illustrates how the image is scanned in to display by the memfetch unit for each combination of (SCAN_COLUMN, H_DIR, V_DIR).

SCAN_COLUMN	H_DIR	V_DIR	Orientation
0	0	0	0 degrees
0	0	1	N/A – vertical flip

SCAN_COLUMN	H_DIR	V_DIR	Orientation
0	1	0	N/A – horizontal flip
0	1	1	180 degrees
1	0	0	N/A – vertical flip followed by rotation
1	0	1	90 degrees
1	1	0	N/A – horizontal flip followed by rotation
1	1	1	270 degrees

The window dimension registers can be confusing when rotation is introduced. Therefore it is important to note which registers apply to pre-rotation and which to post rotation.

The following registers apply to pre-rotation:

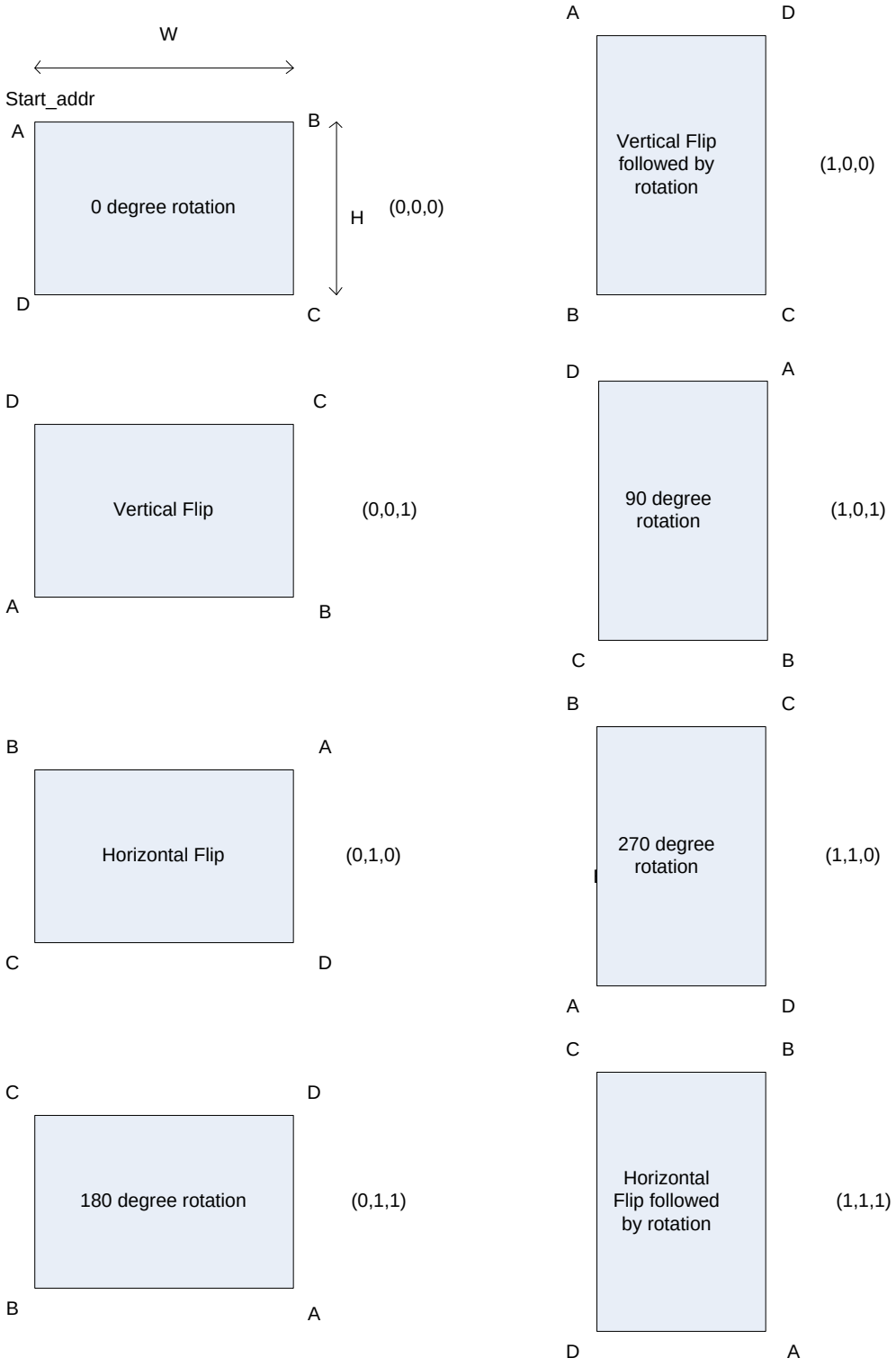
- B_WIN_OPTIONS
  - H_DIR
  - V_DIR
  - SCAN_COLUMN
- B_LINE_STRIDE
- B*_BUF_STRIDE
- B_START_ADDR_*
- B_ ADDR_H_OFFSET_*
- B_ ADDR_V_OFFSET_*

The following registers apply to post rotation:

- B_WIN_OPTIONS
  - H_FILTER*
  - V_FILTER*
  - SCAN_COLUMN
- B_PRESCALED_SIZE
- B_SIZE
- B*_INITIAL_DDA
- B_DDA_INCREMENT

Because the memfetch unit is essentially walking the Y axis in memory, the majority of the window registers and associated logic remain unchanged. Therefore except for B_ADDR_H/V_OFFSET horizontal and vertical directions are with respect to post rotation.

Figure 57: Rotation Combinations



21.3.1.2 Line Buffer

The memfetch line buffer allows the display controller to retain one or more scan lines of window data locally so that no further reading of the scan line is required from memory. The scenarios where this is required are:

- Vertical scaling and filtering
  - Requires 3 scan lines of data for non-rotated memory fetches
  - Requires 5 scan lines of data for rotated memory fetches of packed data
- Reading block linear data
  - Data is sent in 2 scan lines for non-rotated memory fetches
  - Data is sent in 4 scan lines for rotated memory fetches

Data is received from the memory interface and steered to the proper location in the line buffer based on the thread ID in the tag FIFO. Data is read out of the line buffer and unpacked based on the pixel format. These pixels are placed in an output FIFO of memfetch to be sent to the window scaler logic in the display pipe.

In the vertical scaling and filtering scenario, up to five scan lines of data are required for the vertical scaler engine for it to generate the proper scaled output. In the block linear mode scenario, only a single scan line is output but the 32B returned data contains portions of two scan lines. To ensure adequate storage of RGB packed data format and steady bandwidth consumption to the display controller, line buffers are deep enough to hold up to five lines of maximum width of packed pixel data based on the maximum use-case supported by the given window.

Support for YUV planar rotation when reading data in BL 16Bx4 format requires a larger amount of data storage. Each plane of data will return 16B per color component. Due to scaling, it is required to keep a scan line of data in the buffer when fetching the next block of 16. This will require an additional byte of data. Therefore the buffer requirements for YUV planar are:

- Number of bytes per plane = 17 bytes
  - Minimum data fetch size = 16 bytes
  - Scaling requires additional scan-line remain
  - Pixel component depth per plane = 1 byte
- Number of planes = 3
- Line buffer width = 2560 pixels –maximum width
- Line buffer size = 17 bytes/plane* 3 planes/pixel * 2560 pixels per line

  = 130560 bytes

Since the line buffer storage requirement for YUV planar is close to three times (3X) the RGB packed format, the Tegra K1 display only supports buffering support for YUV planar rotation on 1 window. This restriction applies to YUV422 packed data as well due to the implementation of expanding the data to YUV444 in the line buffer. The other two windows will contain buffering to support non YUV planar rotation (51200 bytes).

Below are the equations used to calculate buffer requirements for a given rotated video image.

- YUV444 (packed):
  - Scaling – 5 lines*4Bpp*width
  - No scaling – 4 lines*4Bpp*width
- YUV422 (packed):
  - Scaling – 9 lines*4Bpp(as data is expanded)*width
  - No scaling – 8 lines*4Bpp(as data is expanded)*width
- YUV420 (planar):
  - Scaling: 3 (number of surfaces)*17*width (17 lines @ 1B/line)
  - No scaling: 3 (number of surfaces)*16*width (16 lines @ 1B/line)

The line buffer helps to prevent refetching of data from memory. The line buffer size requirement to prevent refetch depends on the use case. Based on the use case and method of packing/unpacking pixels stored in the line buffer, there will be an

inherent pixel latency associated with the buffer. The memory controller can take advantage of this latency to temporarily stop sending pixel data to the display for cases such as changing the memory clock frequency.

Note that the line buffer can be written or read either with single or multiple lines per clock. For rotation use cases, line buffers load up to 16 lines at a time based on pixel format. Line buffers are read one or two lines at a time based on vertical scaling. The following tables and figures illustrate the line buffer size as well as inherent latency for various use cases for packed ARGB formats (4Bpp). For rotation use cases only block linear formats are supported. Remember that for rotation, block linear format is 16Bx4 lines.

Table 76: Non-Rotation Use Cases

Scenario	Vertical Scaling Enabled	Data Format	Line buffer size required to avoid refetch	Latency Tolerance Guaranteed by Line Buffer
A	NO	Pitched Linear	0	0 lines
B	NO	Block Linear	1 line	0 lines
C	YES	Pitched Linear	1 line	0 lines
D	YES	Block Linear	3 lines	0 lines

Table 77: Rotation Use Cases

Scenario	Vertical Scaling Enabled	Data Format	Line buffer size required to avoid re-fetch	Latency Tolerance Guaranteed by Line Buffer
A	NO	Block Linear	4 line	1 line
B	YES	Block Linear	5 lines	0 lines
C	NO	YUV Planer	16 line	1 line
D	YES	YUV Planer	17 lines	0 lines

The latency tolerance refers to the number of lines available in the line buffer when the line buffer is full between when a given pixel is written to and read from the line buffer. For some of these use cases, there is a relatively long period of time after a pixel is written to the buffer before it is needed in the scan line. However, note that in every case where a scan line or more of data exists, the scan lines are being written at a rate of ½ or ¼ of the pixel clock rate since either 2 or 4 scan lines are being read at once from memory at a rate of 1 pixel per clock.

The fixed size of the Tegra K1 display buffers is given below. Each buffer size has a corresponding “maximum use-case” scenario from those listed above. For those scenarios that are not “maximum use-case”, the portion of the line buffer beyond what the scenario required can be used as latency buffering. Even the “maximum use-case” scenario can use a portion of the pixel latency as latency buffering.

Table 78: Tegra K1 Display Line Buffer Sizes

Line Buffer Size (KB)	Maximum Use-Case	Maximum Resolution
128	Rotation – Scenario C	2560x1600
50	Rotation – Scenario B	2560x1600
48	Non-Rotation – Scenario D	4096x2160

To take advantage of the portion of the line buffer dedicated for handling latency MEMFETCH will send flags to the MC indicating when a programmable threshold is crossed. For each scenario, software must calculate the portion of the line buffer that can be used for latency buffering and set the threshold accordingly. Below is a list of the flags MEMFETCH will generate:

- SPOOLUP
  - Signal name: csr_display0a2mc_spool_up

- Active from when data is requested during Vblank until active space
- Active space starts with 1st pixel of window sent to display pipe

- DVFS readiness flag
  - Requires HWM threshold
  - Signal name: csr_display0a2mc_ready_for_latency_event
  - Active when:

  Buffer exceeds HWM threshold

  Window is disabled

  From the time all data is read for current frame until data is requested for subsequent frame
  - Deactivated with occurrence of underflow for the remainder of the frame
- LATENCY ALLOWANCE SCALE FACTOR buffering
  - Requires HWM/LWM thresholds
  - Signal names:

  csr_display0a2mc_la_th_above_hi = level > HWM

  deactivate with occurrence of underflow for remainder of frame

  activate when window is disabled

  csr_display0a2mc_la_th_below_lo = level < LWM

  activate with occurrence of underflow for remainder of frame

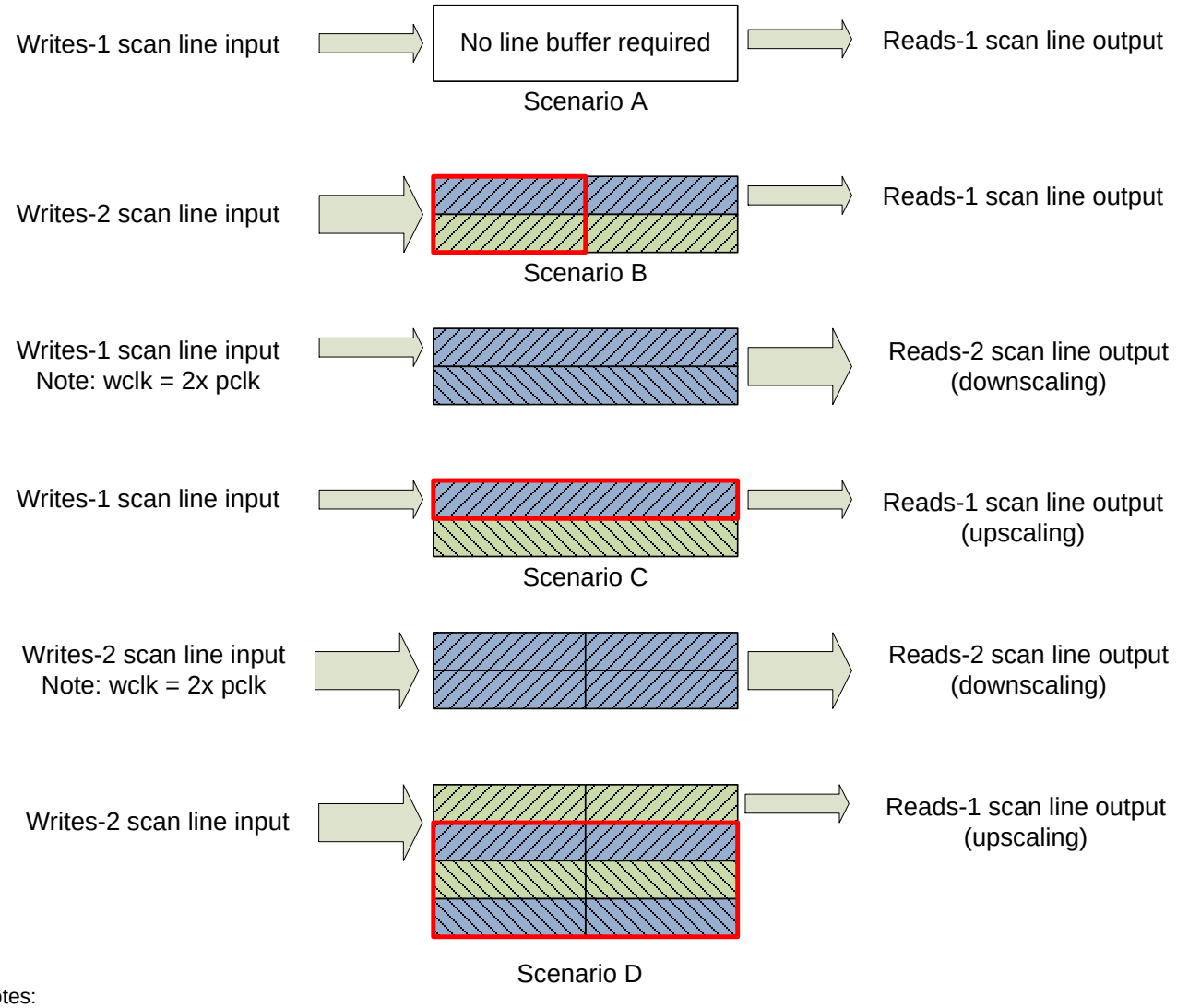
  deactivate when window is disabled

The programmable threshold will simply keep a running count of the number of bytes in the buffer and compare that value against the one in the threshold register.

Line buffering for planer YUV requires separate storage per plane. Each line is read by sending 1KB of Y requests followed by 1KB of U then 1KB of V requests. This is maintained until the entire line of YUV is read. Subsequent line requests are not issued for any component until all component requests have been issued for the current line. MEMFETCH does allow for changing the weights of the given requests per plane to help control the behavior of the requests per plane.

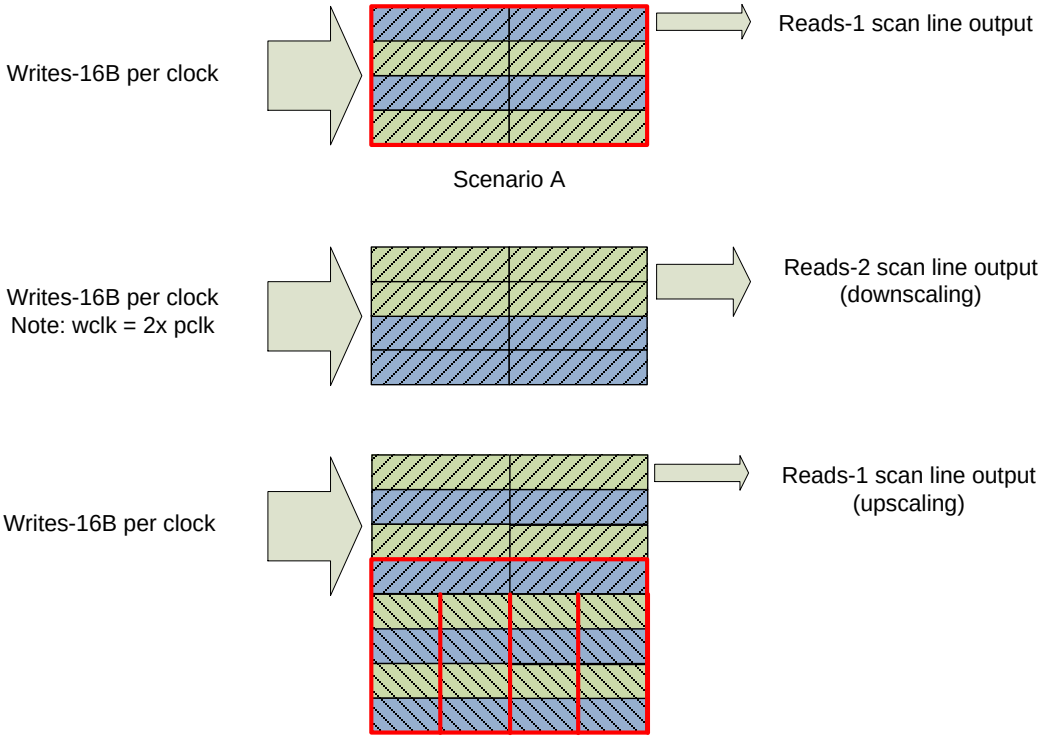
For non-rotated YUV planer use cases, the size of the line buffer is less than that of 4B ARGB since the pixel format is only 1.5B/pixel. However rotated planer YUV requires storage of 16B for each line per plane. Contact your NVIDIA AE should you need further support on this.

Figure 58: Non-Rotation Line Buffer Use Cases



Notes:
- Diagonal Lines represent the order in which data is written to the buffer (Two rows that have the same line pattern are written in parallel)
- Color represents the order in which data is read from the buffer (the same color indicates the rows are read in parallel)
- Red box indicates the line buffer size required to avoid refetch

Figure 59: Rotation Line Buffer Use Cases



Note:
- Diagonal lines Indicate the order in which data is written to the buffer.
  If the two rows have same line pattern, they are written in parallel.
- Color indicates read pattern
- Red boxes indicate line buffer size needed to avoid refetch
- Red lines indicate progress of filling line buffer as scan lines are read

21.3.1.3 Handling of Underflow

Display is an isochronous client that requires a steady bandwidth allocation. If data is not returned at a fairly constant rate and the line buffers run dry then data will not appear on the screen at the proper time. Eventually when data does start to flow to the display controller the image will appear shifted and there is no way for the image data to catch up to where it should be on the screen. This undesirable lack of guaranteed data is called underflow and it can happen in 2 forms:

- Short term underflow – pixels worth of missing data
- Long term underflow – scan-lines worth of missing data

For short term underflow it is possible for the display controller to catch up to the correct pixel by the start of the next scan-line by reading pixels during the horizontal blanking period. Short term underflow is being managed by Tegra display controller in this manner. For long term underflow it would be impossible for the display controller to throw out that much data. Short term underflow may arise when the memory controller is over-taxed based on a momentary surge in memory client requests. Long term underflow may happen when memory controller frequency changes are being made.

Tegra K1 display windows will address short term underflow similar to what is done in Tegra 3. Long term underflow is also handled in a manner similar to Tegra 3 during the active frame portion. Below is a description of Tegra K1 Underflow recovery improvements:

- Underflow recovery within a frame

- Start counter of underflow pixels when underflow occurs
- Last valid pixel is held at the beginning at the display pipeline for duration of the underflow
- Underflow pixels that eventually return to display are stored in line buffer and immediately read/discarded
- Operation continues (even during HBlank) until all underflow pixels have been discarded
- Once valid pixel is present at display pipe input memfetch operations return to normal mode
- Limit requests issued at the end of a frame if window is in underflow using the following method:

  For cases when underflow continues to the end of the frame

  Add threshold counter (B_UFLOW_THRESHOLD)

  Track prescaled scanout

  Stop issuing frame requests when

  prescaled scanout + threshold is > prescaled size

- Underflow recovery between frames
  - Add a frame bit to the tag FIFO to allow DP logic to determine which of the responses that are coming back are for the previous frame
  - Drop pixels from previous frame at 64B/pclk rate
  - Within display pipe, reset the underflowed pixel counters
  - Limit requests issued at the end of a frame if window is in underflow using method described above
  - Last pixel of previous frame is displayed

The Tegra K1 display handles cursor underflow in the following manner:

- Interface between memfetch and display pipeline senses when underflow occurs.
- For simple windows, the display pipe drives the last valid piel for the duration of the underflow.
- When underflow occurs display pipeline drives a cursor alpha value of 0 for the remainder of the underflow period. This could be until the end of 1 or more scan lines.
- Memfetch will not drive valid to the display pipeline until the start of the scan line when data is available.
- Display pipeline will not assert ready to memfetch until the beginning of the scan line when data is available and memfetch is driving valid.
- Memfetch keeps track of number of clocks of missing data and when data does appear it reads out the data until the end of the scan line is reached.

It is important to highlight underflows for debug purposes. Current display units raise an interrupt to indicate that underflow has occurred. The Tegra K1 display continues to drive an interrupt when an underflow is detected. To further assist with highlighting underflows, memfetch drives a solid color from the window that experiences an underflow if the associated enable bit is active.

21.3.2 Interlaced Fields

To support set top boxes, the Tegra K1 display supports interlaced format, which is still prevalent in many television displays and broadcast networks. Interlaced format requires half the bandwidth of progressive and alternates sending fields containing odd and even lines of the original video frame.

The CEA Standard (CEA-861-E) defines these interlaced fields as FIELD1 and FIELD2. FIELD1 contains the even lines (line 0, 2, 4, etc.). FIELD2 contains the odd lines (line 1, 3, 5, etc.). From a consumer electronic display perspective, FIELD1 contains the top lines, and FIELD2 contains the bottom lines.

Memfetch contains start (base) addresses for both fields and when interlaced mode is enabled it will toggle between these base addresses every frame. Memfetch will continue to toggle between FIELD1 and FIELD2 fields until interlace mode is disabled. When interlace mode is disabled, memfetch only uses the normal (FIELD1) registers.

IMPORTANT: The first field that is fetched when interlace mode is based on HVTGEN field1/field2 field status. It is up to software to ensure that proper field being fetched is ready when activating interlace mode in memfetch.

No parameter other than start address and offset can change between fields such as size and stride.

To support double buffering of interlaced fields, software is required to update field1/field2 base addresses on a frame (not a field) boundary. Becausse the display is treating each field as a frame, software should make sure that updates only occur after an even number of frame SYNCPTS are received. It is strongly suggested that both field1 and field2 fields are ready in memory when performing the buffer swap to avoid coordinating field updates based on current field scanout.

21.3.3 Surface/Color Formats

In general, the Tegra K1 display supports the same pixel and surface formats used in Tegra 3. These pixel formats are not supported by the Tegra K1 display controller:

- P1 – 1bpp palletized
- P2 – 2bpp palletized
- P4 – 4bpp palletized
- RGB666
- YUV422RA

Below are the surface and pixel formats supported by the Tegra K1 display controller. Formats are lsb-to-msb.

Surface formats:

- Pitched linear – scan-line data
- Block linear
  - 32B x 2 line – rotation off
  - 16B x 4 line – rotation on

Pixel formats:

- RGB packed
  - T_P8 – single 8-bit color index component. RGB value is looked up in LUT.
  - T_R5G6B5, T_B5G6R5 – 5 bits for R/B, 6 bits for G.
  - T_A4R4G4B4, T_A4B4G4R4 – 4 bits each, packed into 16-bit word. Each is expanded to 8 bits by MSB replication or 0-fill, depending on COLOR_EXPAND.
  - T_A1R5G5B5, T_R5G5B5A1, T_A1B5G5R5, T_B5G5R5A1: 5 bits each for RGB, 1 bit for alpha. Expanded to 8 bits by MSB replication or 0-fill, depending on COLOR_EXPAND.
  - T_A1R5G5B5, T_R5G5B5A1, T_A1B5G5R5, T_B5G5R5A1: 5 bits each for RGB, 1 bit for alpha. Expanded to 8 bits by MSB replication or 0-fill, depending on COLOR_EXPAND.
  - T_X1R5G5B5, T_R5G5B5X1, T_X1B5G5R5, B5G5R5X1: 5 bits for RGB Expanded to 8 bits by MSB replication or 0-fill, depending on COLOR_EXPAND.
  - T_A8R8G8B8, T_R8G8B8A8, T_A8B8G8R8, T_B8G8R8A8: 8 bits for R/G/B/A
  - T_X8R8G8B8, T_R8G8B8X8, T_X8B8G8R8, T_B8G8R8X8: 8 bits for R/G/B

- YUV packed
  - T_Y8_U8__Y8_V8, T_Y8_V8__Y8_U8, T_U8_Y8__V8_Y8, T_V8_Y8__U8_Y8: Cb and Cr are stored decimated 2:1 horizontally, yielding two bytes per pixel: Y0 Cb0 Y1 Cr0 Y2 Cb2 Y3 Cr 2 …. Cb and Cr must be replicated to generate YCbCr444 for use in display pipeline.
- YUV planar
  - T_Y8___U8___V8_N420: (YCbCr420P) Cb and Cr are stored decimated 2:1 horizontally and vertically, in separate surface planes. Cb and Cr surfaces are ¼ size of Y plane. Each Cb and Cr pixel is shared (replicated) by 4 pixels. To avoid fetching Cb and Cr lines twice for each Y line, a line buffer is needed to cache Cb and Cr.
  - T_Y8___U8___V8_N422: (YCbCr422P) Cb and Cr are stored decimated 2:1 horizontally, in separate surface planes. Cb and Cr surfaces are ½ size of Y plane. Each Cb and Cr pixel is shared (replicated) by 2 pixels.
  - T_Y8___U8___V8_N444: 4:4:4 planar.
- Semi- planar
  - T_Y8___U8V8_N420, T_Y8___V8U8_N420:(YCbCr420 semi-planar) Cb and Cr are stored decimated 4:1 horizontally, interleaved in a CbCr plane. CbCr surface is ½ size of Y plane. Each Cb and Cr pixel is shared (replicated) by 4 pixels. Y plane has: Y0 Y1 Y2 … and CbCr plane has: Cb0 Cr0 Cb4 Cr4 …
  - T_Y8___U8V8_N422, T_Y8___V8U8_N422:(YCbCr422 semi-planar) Cb and Cr are stored decimated 2:1 horizontally, interleaved in a CbCr plane. CbCr surface is same size as Y plane. Each Cb and Cr pixel is shared (replicated) by 2 pixels. Y plane has: Y0 Y1 Y2 … and CbCr plane has: Cb0 Cr0 Cb2 Cr2 …
  - T_Y8___U8V8_N444, T_Y8___V8U8_N444: 4:4:4 semiplanar.

True YUV formats are supported in display along with YCrCb. YUV differs from YCbCr in that UV is signed -127 to 128 where CbCr is offset 128 and ranges 0 to 255.

The following formats are reserved for true YUV as opposed to YCbCr:

- T_A8Y8U8V8_TRUE, T_V8U8Y8A8_TRUE
- T_U8_Y8__V8_Y8_TRUE
- T_Y8___U8___V8_N420_TRUE
- T_Y8___U8___V8_N422_TRUE
- T_Y8___U8___V8_N422R_TRUE
- T_Y8___U8___V8_N444_TRUE
- T_Y8___U8V8_N420_TRUE
- T_Y8___V8U8_N420_TRUE
- T_Y8___U8V8_N422_TRUE
- T_Y8___V8U8_N422_TRUE
- T_Y8___U8V8_N422R_TRUE
- T_Y8___V8U8_N422R_TRUE
- T_Y8___U8V8_N444_TRUE
- T_Y8___V8U8_N444_TRUE

The Tegra K1 display supports a subset of the above pixel formats natively. To support all display pixel formatsrequires use of the window B_BYTE_SWAP register. The table below gives a full list of the required display pixel formats along with the display native format and B_BYTE_SWAP settings.

Format	Color_Depth	Byte_Swap
T_B5G6R5	T_B5G6R5	NA
T_R5G6B5	T_R5G6B5	NA

Format	Color_Depth	Byte_Swap
T_A1B5G5R5	T_A1B5G5R5	NA
T_A1R5G5B5	T_A1R5G5B5	NA
T_B5G5R5A1	T_B5G5R5A1	NA
T_R5G5B5A1	T_R5G5B5A1	NA
T_X1B5G5R5	T_X1B5G5R5	NA
T_X1R5G5B5	T_X1R5G5B5	NA
T_B5G5R5X1	T_B5G5R5X1	NA
T_R5G5B5X1	T_R5G5B5X1	NA
T_A4B4G4R4	T_A4B4G4R4	NA
T_A4R4G4B4	T_A4R4G4B4	NA
T_A8B8G8R8	T_A8B8G8R8	NA
T_A8R8G8B8	T_A8R8G8B8	NA
T_B8G8R8A8	T_A8R8G8B8	SWAP4
T_R8G8B8A8	T_A8B8G8R8	SWAP4
T_X8B8G8R8	T_X8B8G8R8	NA
T_X8R8G8B8	T_X8R8G8B8	NA
T_B8G8R8X8	T_X8R8G8B8	SWAP4
T_R8G8B8X8	T_X8B8G8R8	SWAP4
T_P8	T_P8	NA
T_Y8_U8__Y8_V8 (YUV422 packed)	T_U8_Y8__V8_Y8	SWAP2
T_Y8_V8__Y8_U8 (YUV422 packed)	T_U8_Y8__V8_Y8	SWAPLEFT
T_U8_Y8__V8_Y8 (YUV422 packed)	T_U8_Y8__V8_Y8	NA
T_V8_Y8__U8_Y8 (YUV422 packed)	T_U8_Y8__V8_Y8	SWAP02
T_Y8___U8V8_N444 (YUV444 semi-planar)	T_Y8___U8V8_N444	NA
T_Y8___V8U8_N444 (YUV444 semi-planar)	T_Y8___V8U8_N444	NA
T_Y8___U8V8_N422 (YUV422 semi-planar)	T_Y8___U8V8_N422	NA
T_Y8___V8U8_N422 (YUV422 semi-planar)	T_Y8___V8U8_N422	NA
T_Y8___U8V8_N422R (YUV422R semi-planar)	T_Y8___U8V8_N422R	NA
T_Y8___V8U8_N422R (YUV422R semi-planar)	T_Y8___V8U8_N422R	NA
T_Y8___U8V8_N420 (YUV420 semi-planar)	T_Y8___U8V8_N420	NA
T_Y8___V8U8_N420 (YUV420 semi-planar)	T_Y8___V8U8_N420	NA
T_Y8___U8___V8_N444 (YUV444 planar)	T_Y8___U8___V8_N444	NA
T_Y8___U8___V8_N422 (YUV422 planar)	T_Y8___U8___V8_N422	NA
T_Y8___U8___V8_N422R (YUV422R planar)	T_Y8___U8___V8_N422R	NA

Format	Color_Depth	Byte_Swap
T_Y8___U8___V8_N420 (YUV420 planar)	T_Y8___U8___V8_N420	NA

21.3.4 Horizontal Filter

Memfetch and scaler provide 6 adjacent pixels per clock, along with DDA fraction indicating filter kernel phase. Fraction selects among 16 sets of coefficients for filter kernel.

21.3.5 Vertical Filter

When enabled, memfetch provides 2 adjacent lines in parallel, which are independently horizontally filtered before being sent to vertical filter. The fractional DDA from vertical scaler (indicating filter phase) selects among 16 filter kernel coefficients. For coefficient entry N, the weights are N and 1-N.

21.3.6 Color Space Converter

Most of the display datapath is RGB. The color space conversion unit can convert YCbCr and YUV to RGB. It can also do some RGB to RGB conversions.

21.3.7 Color Palette/LUT

To support gamma and palletized formats, the LUT has three 256x8-bit tables, one each for red, green, blue. The three tables are independent and can represent arbitrary functions. For 4-bit palette modes, only the bottom 16 entries are used.

21.3.8 Digital Vibrance

Using the same algorithm as NVIDIA GeForce GPUs, digital vibrance can increase the saturation of colors.

21.3.9 Cursor

The cursor supports two pixel formats:

- Legacy 2-bit-per-pixel, where the four values are background color, foreground color, transparent, and invert. Foreground and background color are programmable via registers with 24-bit RGB values. Transparent means cursor is invisible and desktop shows through. Invert means desktop RGB is inverted (red' = 255 – red, etc.). Pixels are packed 4 per byte, and each line has 16 bytes pitch, regardless of whether it is 32 or 64 pixels wide.
  - Cursor sizes of 32x32 or 64x64
- 32-bpp RGBA pitch linear
  - Full color with alpha
  - Cursor sizes of:
    - 32x32
    - 64x64
    - 128x128
    - 256x256

The blending required for the alpha cursor will occur after the blending stage. It uses the following blend equation:

Final_image = cursor_alpha * cursor_color + (1-cursor_alpha) * blending_stage_output_color

Note: Legacy cursor does not support per-pixel or global alpha.

Cursor can be clipped to either the display, or window A, B, or C. Thus the cursor active region for blending must be computed according to that mode. In regions where cursor is clipped, or outside the cursor entirely, blending behaves as if cursor is disabled (i.e., previous blend stage passes through.)

> **Note:** The Tegra K1 display cursor supports partial clipping but does not support trivial reject of cursor when the cursor is completely outside of the window region. It is the responsibility of software to ensure that the cursor is at least partially visible.

21.3.9.1 Legacy Support for Blending

The Tegra K1 display supports cursor modes used in previous generations of the Tegra display. Legacy cursor blending uses 2 bits per pixel and programming for this mode will remain unchanged.

21.3.9.2 32-bpp RGBA Cursor Mode

In 32-bpp RGBA cursor mode, legacy adaption is disabled. Cursor RGBA is blended with the last stage of the window blending unit. Cursor alpha value controls this blending.

21.3.9.3 Interlaced Cursor Support

To support interlaced output, the cursor performs a progressive-to-interlaced format conversion of the cursor image in memory. Since the cursor is static, no low pass filtering is necessary. The display cursor reads every other line from the square cursor thus reading only half of the lines. Based on the cursor destination Y position and the field, the Tegra K1 display starts reading line 0 or 1 from memory. The decision on whether to start on cursor line 0 or 1 for the first or second field is left to software, giving more flexibility to manipulate the cursor image and set the offset for a given field. An interlace sequence example for cursor is shown below:

1. Program CURSOR_INTERLACE_ENABLE=1
2. Program CURSOR_INTERLACE_START =1
3. The cursor reads every other line from memory starting at line 1 and stores it to sequential panel lines
4. On the next “frame”, the cursor toggles the line start to 0

   CURSOR_INTERLACE_STATUS = 0
5. The cursor reads every other line from memory starting at line 0 and stores it to sequential panel lines

Repeat steps 2 through 5 in the above sequence when the cursor position is updated and set the cursor start line appropriately.

The cursor interlace control register is shown below. When the CURSOR _INTERLACE_START bit is written, hardware should update the corresponding CURSOR_INTERLACE_STATUS bit to this value. This allows the start line to track updates to the cursor position. The CURSOR_INTERLACE_FIELD1/2_VOFF_INCR bits determine if V_CURSOR_POSITION is incremented by one for that field. The field status is tracked by the hardware cursor logic using HVTGEN interlace field status information.

Table 79: Cursor Interlace Control Register

Field Name	Setting
CURSOR_INTERLACE_ENABLE	Interlace Enable 0=disable 1=enable
CURSOR_INTERLACE_FIELD1_VOFF_INCR	Field1 v position Incr Enable 0=disable 1=enable

Field Name	Setting
CURSOR_INTERLACE_FIELD2_VOFF_INCR	Field2 v position Incr Enable 0=disable 1=enable
CURSOR _INTERLACE_START	Starting line. 0=line 0, 1=line 1
CURSOR_INTERLACE_STATUS	Current line start. Read only. 0=line 0, 1=line 1

There are two possibilities for cursor vertical position in a progressive frame – odd or even line. When breaking a progressive frame into interlaced fields, it is important to know what cursor line number to start on per field when the cursor vertical position starts on an odd progressive line and when it is on an even progressive line. Consider the following examples:

Example 1 cursor starts on line 10:

- V_CURSOR_POSITION = 5
- First line of cursor is at progressive line offset 10

  Corresponds to FIELD1 line 5
- Second line of cursor is at progressive line offset 11

  Corresponds to FIELD2 line 5

Example 2 cursor starts on line 1:

- V_CURSOR_POSITION = 0
- First line of cursor is at progressive line offset 1

  Corresponds to FIELD2 line 0
- Second line of cursor is at progressive line offset 2

  Corresponds to FIELD1 line 1

  Requires setting CURSOR_INTERLACE_FIELD1_VOFF_INCR=1

The following table describes programming mechanism for line start based on progressive frame vertical cursor offset for interlaced fields FIELD1 and FIELD2.

Interlace field	Progressive frame cursor Y offset	
	Odd	Even
First field (FIELD1)	CURSOR_INTERLACE_START = Line 1 V_CURSOR_POSITION = INT(Progressive_cursor_y/2) + 1	CURSOR_INTERLACE_START = Line 0 V_CURSOR_POSITION = INT(Progressive_cursor_y/2)
Second field (FIELD2)	CURSOR_INTERLACE_START = Line 0 V_CURSOR_POSITION = INT(Progressive_cursor_y/2)	CURSOR_INTERLACE_START = Line 1 V_CURSOR_POSITION = INT(Progressive_cursor_y/2)

Note: Programming of CURSOR_INTERLACE registers listed above must line up with the interlaced field in the table. Therefore the register should be written (armed) on the previous field for it to activate on the appropriate field.

21.3.10 Blending

The Tegra K1 display uses weighted blending between windows. Legacy cursor blending must be disabled when full-color alpha-blended cursor is in use.

In Tegra K1 devices, the display blender has six sequential blending stages. Each window is assigned a depth value. The windows are sorted by depth and assigned to a blend stage. The cursor is the last (top) stage. Each stage combines its pixels with the previous stage according to programmed equations.

Figure 60: Tegra K1 Blending

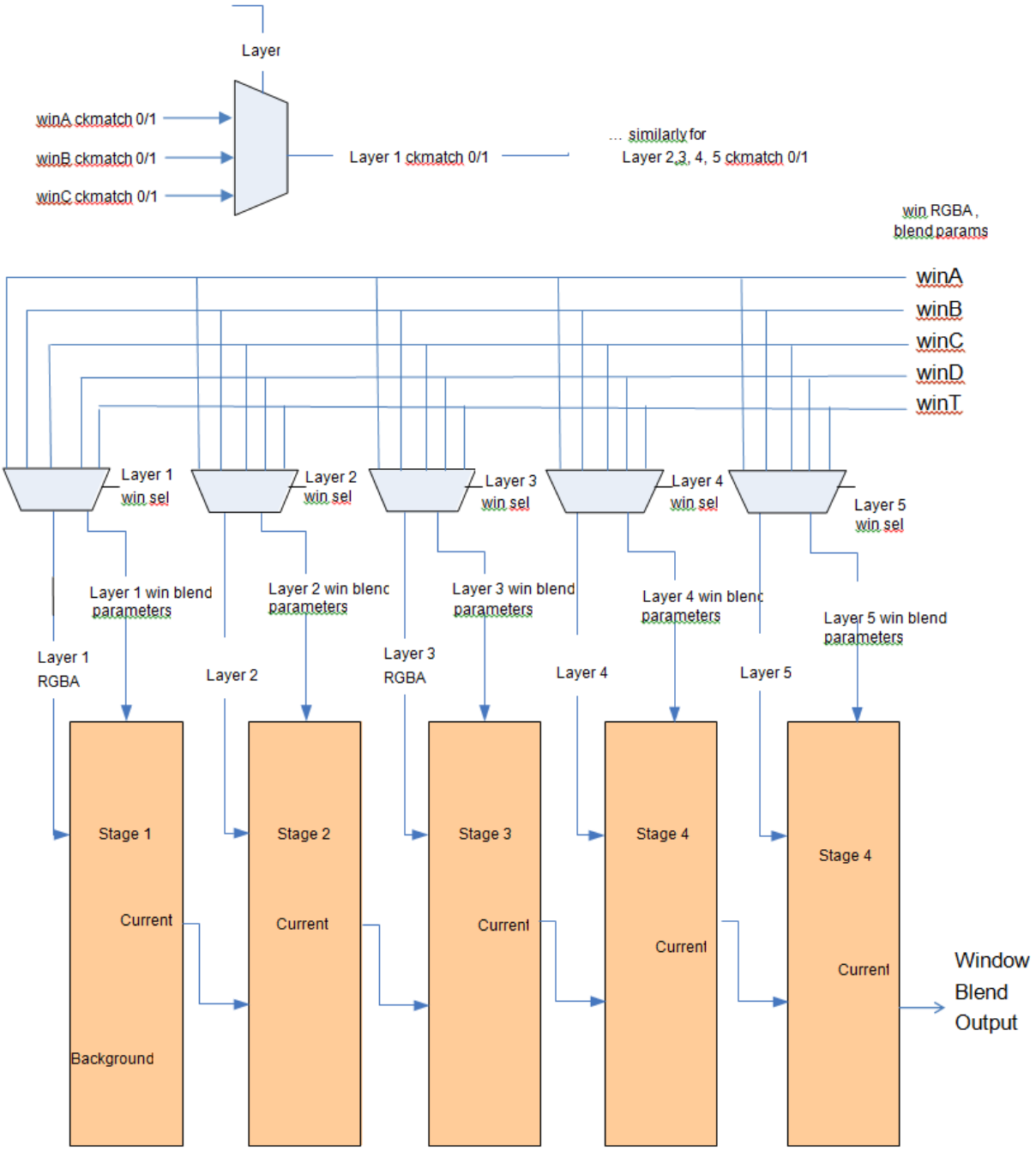
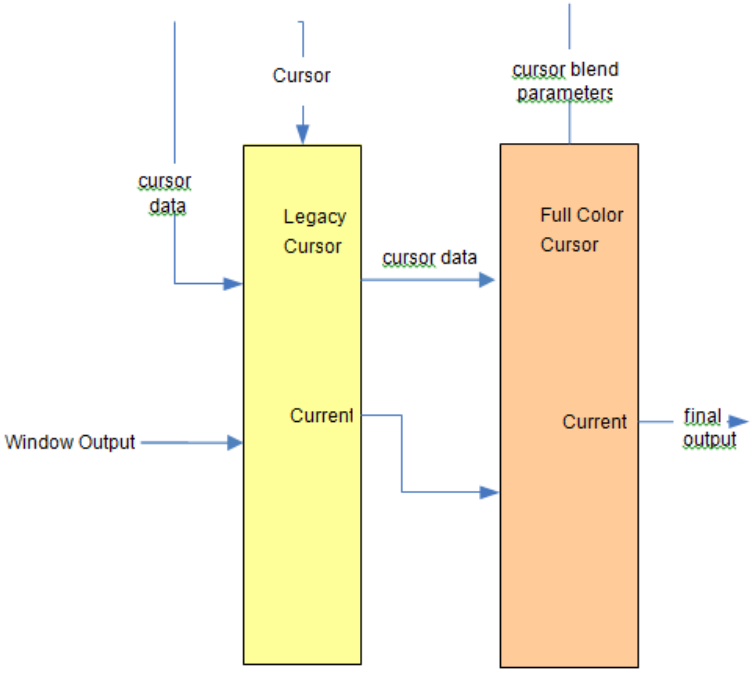


Figure 61: Tegra K1 Blender Cursor Stages



21.3.10.1 Blend Algorithm

Each blending stage requires:

- The current color from the previous stage (often called “Destination”)
- The surface color from the assigned window (often called “Source”)
- Window enable (that is, whether windows cover the current pixel)
- Blend parameters from the per-window registers

Each window has the following registers (where x is A, B, C, D, or T). The cursor settings are also described below.

```
windowx.Z  - an 8 bit z value
windowx.K1 - an 8 bit constant alpha value
windowx.K2 - an 8 bit constant alpha value
windowx.blend_bypass - a boolean

windowx.srcFactC_Match_Select - an enum with the following options:
            K1                      windowx.K1
            K1_TIMES_DST            windowx.K1*currentA
            NEG_K1_TIMES_DST        1 - (windowx.K1*currentA)
            K1_TIMES_SRC            windowx.K1*layerxA
            ZERO                    0

windowx.dstFactC_Match_Select - an enum with the following options:
            K1                      windowx.K1
            K2                      windowx.K2
            K1_TIMES_DST            windowx.K1*currentA
```

```
                NEG_K1_TIMES_DST        1 - (windowx.K1*currentA)
                NEG_K1_TIMES_SRC        1 - (windowx.K1*layerxA)
                ZERO                    0
                ONE                     1.0 (0xff)

    windowx.srcFactA_Match_Select - an enum with the following options:
                K1                      windowx.K2
                K2                      windowx.K2
                ZERO                    0

    windowx.dstFactA_Match_Select - an enum with the following options:
                K2                      windowx.K2
                ZERO                    0
                NEG_K1_TIMES_SRC        1 - (windowx.K1*layerxA)
                ONE                     1.0 (0xff)
```

Tegra K1 devices support color key; however, color key is not described in this document. The “Match” blend registers above apply to cases where color key is disabled as well as to cases where color key is enabled but the key color does not match. There is an additional set of “NoMatch” registers for the color key enabled but not matching cases.

The windows are sorted according to Z (Depth) value and assigned to blending stages 1 to 5. Cursor is layer 6. Each layerx (where x is 1 .. 6) operates as follows:

Inputs

```
    layerxRGBA - the RGBA color from the corresponding window pipeline
    windowx.* - window registers from window associated with this layer
    currentxRGBA - output from the preceding layer_blendx unit
```

Layer0 only needs the BackgroundColorRGBA (also known as Border Color) as input

Layer 6 (cursor) gets layerxRGBA from the cursor surface and gets windowx.* from special cursor.* registers as follows:

```
                windowx.key_Select = NONE
                windowx.K1 = cursor.K1
                windowx.K2 = cursor.K2
                windowx.dstFactC_Match_Select = cursor.dstFactC_Select
                windowx.srcFactC_Match_Select = cursor.srcFactC_Select
```

The following are “don’t cares” but can be set as shown:

```
                windowx.dstFactA_Match_Select   = K2
                windowx.srcFactA_Match_Select   = K2
```

Outputs

```
    outputRGBA – 32-bit blended RGBA color
```

Local State

```
    match - boolean
    dstFactC_Select - enum
    srcFactC_Select - enum
    dstFactA_Select - enum
    srcFactA_Select - enum
    dstFactC - 8 bit alpha value
    srcFactC - 8 bit alpha value
    dstFactA - 8 bit alpha value
    srcFactA - 8 bit alpha value
```

Operation

At each pixel of the output raster, the blending is performed. Only a subset of windows touch (cover) each pixel, because windows have an arbitrary size and position on the “desktop” raster.

layer_blend0 is trivial:

```
        outputRGBA = BackgroundColorRGBA
```

layer_blendx (where x is 1 through 5) does the following:

```
    - calculate fact??_Select:
        if [match] then
            srcFactC_Select = windowx.srcFactC_Match_Select
            dstFactC_Select = windowx.dstFactC_Match_Select
            srcFactA_Select = windowx.srcFactA_Match_Select
            dstFactA_Select = windowx.dstFactA_Match_Select
        else
            srcFactC_Select = windowx.srcFactC_NoMatch_Select
            dstFactC_Select = windowx.dstFactC_NoMatch_Select
            srcFactA_Select = windowx.srcFactA_NoMatch_Select
            dstFactA_Select = windowx.dstFactA_NoMatch_Select

    - calculate srcFactC:
        switch (srcFactC_Select) {
            case K1:                    srcFactC = windowx.K1
            case K1_TIMES_SRC:          srcFactC = windowx.K1*layerxA
            case K1_TIMES_DST:          srcFactC = windowx.K1*currentA
            case NEG_K1_TIMES_DST:      srcFactC = 1 - (windowx.K1*currentA)
            case ZERO:                  srcFactC = 0
            case ONE:                   srcFactC = 0
        }

    - calculate dstFactC:
        switch (dstFactC_Select) {
            case K1:                    dstFactC = windowx.K1
            case K2:                    dstFactC = windowx.K2
            case ZERO:                  dstFactC = 0
            case ONE:                   srcFactC = 0xff
            case K1_TIMES_DST:          dstFactC = windowx.K1*currentA
            case NEG_K1_TIMES_SRC:      dstFactC = 1 - (windowx.K1*layerxA)
            case NEG_K1_TIMES_DST:      dstFactC = 1 - (windowx.K1*currentA)
            case NEG_K1:                dstFactC = 1 - windowx.K1
        }

    - calculate srcFactA:
        switch (srcFactA_Select) {
            case K1:                    srcFactA = windowx.K1
            case K2:                    srcFactA = windowx.K2
            case ZERO:                  srcFactA = 0
        }

    - calculate dstFactA:
        switch (dstFactA_Select) {
            case K2:                    dstFactA = windowx.K2
            case ZERO:                  dstFactA = 0
            case ONE:                   dstFactA = 1
```

```
            case NEG_K1_TIMES_SRC:      dstFactA = 1 - (windowx.K1*layerxA)
        }
```

If the window or cursor for this layer does NOT touch this pixel, then:

```
            outputRGBA = currentRGBA
        else if windowx.blend_bypass is TRUE then
            outputRGBA = layerxRGBA
        else
            outputR = (srcFactC * layerxR) + (dstFactC * currentR)
            outputG = (srcFactC * layerxG) + (dstFactC * currentG)
            outputB = (srcFactC * layerxB) + (dstFactC * currentB)
            outputA = (srcFactA * layerxA) + (dstFactA * currentA)
```

Note: windowx.blend_bypass allows bypassing the entire blend operation and is intended to save power.

For the cursor layer, there is no need to calculate the following:

```
        outputA - not needed
        dstFactA - not needed
        srcFactA - not needed
```

Final Result

The outputRGB from layer_blend6 is the pixel value sent to the display.

21.3.10.2 Example Use Cases

The examples below assume 3 windows, not 5, for simplicity.

```
                Eye is looking down from here
                            i
                            ^
    layer 4                -----              cursor
    layer 3             -------               windowC
    layer 2                  ----------       windowB
    layer 1          ----------------------    windowA
    layer 0          ----------------------    background
```

Common Settings

If not mentioned in a use case, assume registers have the following values:

```
    BackgroundColorRGBA = background color

    key0.MinRGBA = (don't care)
    key0.MaxRGBA = (don't care)

    key1.MinRGBA = (don't care)
    key1.MaxRGBA = (don't care)

    windowA.Z  = 3
    windowB.Z  = 2
    windowC.Z  = 1

    window*.K1 = 0
    window*.K2 = 0
    window*.key_Select = NONE
    window*.dstFactC_Match_Select = K1
    window*.srcFactC_Match_Select = K1
```

```
    window*.dstFactC_NoMatch_Select = K1
    window*.srcFactC_NoMatch_Select = K1
    window*.dstFactA_Match_Select = K2
    window*.srcFactA_Match_Select = K2
    window*.dstFactA_NoMatch_Select = K2
    window*.srcFactA_NoMatch_Select = K2
```

If the cursor uses non-premultiplied alpha with fade factor F:

```
        cursor.K1 = F
        cursor.dstFactC_Select = NEG_K1_TIMES_SRC
        cursor.srcFactC_Select = K1_TIMES_SRC

        result = ((1.0-(F*cursorA)) * layer3RGB) + (F*cursorA * cursorRGB)
```

If the cursor uses premultiplied alpha with fade factor F:

```
        cursor.K1 = F
        cursor.dstFactC_Select = NEG_K1_TIMES_SRC
        cursor.srcFactC_Select = K1

        result = ((1.0-(F*cursorA)) * layer3RGB) + (F         * cursorRGB)
```

Below "use for fade" indicates that the value of windowx.K1 can be used to fade the window from invisible (transparent) at windowx.K1=0x00 to fully visible at windowx.K1=0xff.

Alpha Blend

```
   layer 4               -----              cursor
   layer 3           ccccccc                windowC non-premul
   layer 2                bbbbbbbbbb        windowB premul
   layer 1         aaaaaaaaaaaaaaaaaaaaa    windowA no alpha
   layer 0        gggggggggggggggggggggggg  background

   result         gaaaccccccbbbbbbbbaaaag
                        ^   ^ ^
                        |   | |
                        |   | +---- b blended with a
                        |   +------ c blended with (b blended with a)
                        +---------- c blended with a
```

Window A has RGB (alpha not used)

Window B has premultiplied alpha RGBA for blending with A

Window C has non-premultiplied alpha RGBA for blending with A and B

Blend Equations

```
layer0RGBA = BackgroundColorRGBA
layer1RGB  = windowA color
layer1A    = don't care
layer2RGB  =             windowB.RGB  +  (1.0 - windowB.A) * layer1RGB
layer2A    = don't care
layer3RGB  = windowC.A * windowC.RGB  +  (1.0 - windowC.A) * layer2RGB
layer3A    = don't care
layer4RGB  = blend with cursor
```

Register Settings

```
windowA.K1 = 0xff  (use for fade)
windowA.key_Select = NONE
windowA.srcFactC_Match_Select = K1
windowA.dstFactC_Match_Select = ZERO

windowB.K1 = 0xff  (use for fade)
windowB.key_Select = NONE
windowB.srcFactC_Match_Select = K1  (K1_TIMES_SRC for non-premul)
windowB.dstFactC_Match_Select = NEG_K1_TIMES_SRC

windowC.K1 = 0xff  (use for fade)
windowC.key_Select = NONE
windowC.srcFactC_Match_Select = K1_TIMES_SRC    (K1 for premul)
windowC.dstFactC_Match_Select = NEG_K1_TIMES_SRC
```

Note: Window B and window C can each be either premul or non-premul

Use windowx.K1 to fade the window (0=transparent, 0xff=opaque)

Opaque Layer with Fade

```
layer 4               -----             cursor
layer 3           ccccccc               windowC non-premul
layer 2                bbbbbbbbbb       windowB no alpha
layer 1        aaaaaaaaaaaaaaaaaaaaa    windowA no alpha
layer 0       gggggggggggggggggggggggg  background

result        gaaacccccccbbbbbbbbaaaag
                    ^   ^ ^
                    |   | |
                    |   | +---- b blended with a using L
                    |   +------ c blended with (b blended with a using L)
                    +---------- c blended with a
```

Same as case 1, but window B (or A or C or any combination) contains no alpha values (or garbage alpha values). There is a constant L which is used to fade the window.

Blend Equations

Note: "..." means "same as the last use case"

```
...
layer2RGB  = L * windowB.RGB  +  (1.0 - L) * layer1RGB
layer2A    = don't care
...
```

Register Settings

```
...
windowB.K1 = L
windowB.K2 = 1.0 - L
windowB.srcFactC_Match_Select = K1
windowB.dstFactC_Match_Select = K2
...
```

Window B's Alpha is in Window A

```
layer 4               -----             cursor
layer 3           ccccccc               windowC premul
layer 2                bbbbbbbbbb       windowB no alpha
layer 1        aaaaaaaaaaaaaaaaaaaaa    windowA alpha for blend w/ b
layer 0       gggggggggggggggggggggg    background

result        gaaacccccccbbbbbbbbaaaag
                    ^   ^ ^
                    |   | |
                    |   | +---- b blended with a using alpha from a
                    |   +------ c blended with (")
                    +---------- c blended with a
```

Same as case 1, but window B contains no alpha values (or garbage alpha values), and the alpha is needed for use from window A to blend window B with window A.

Blend Equations

```
layer0RGBA = BackgroundColorRGBA
layer1RGB  = windowA color
layer1A    = windowA alpha
layer2RGB  = windowA.A * windowB.RGB + (1.0 - windowA.A) * layer1RGB
layer2A    = don't care
layer3RGB  = windowC.RGB + (1.0 - windowC.A) * layer2RGB
layer3A    = don't care
layer4RGB  = blend with cursor
```

Register Settings

```
   windowA.K1 = 0xff  (use for fade)
   windowA.K2 = 0xff
   windowA.key_Select = NONE
   windowA.srcFactC_Match_Select = K1
   windowA.dstFactC_Match_Select = ZERO
   windowA.srcFactA_Match_Select = K2
   windowA.dstFactA_Match_Select = ZERO

   windowB.K1 = 0xff  (use for fade)
   windowB.key_Select = NONE
   windowB.srcFactC_Match_Select = NEG_K1_TIMES_DST
   windowB.dstFactC_Match_Select = K1_TIMES_DST

   windowC.K1 = 0xff  (use for fade)
   windowC.key_Select = NONE
   windowC.srcFactC_Match_Select = K1  (K1_TIMES_SRC for non-premul)
   windowC.dstFactC_Match_Select = NEG_K1_TIMES_SRC
```

There is no way to do this for B and C at the same time (unless they are known to not overlap).

If windowA.A is the opacity of window C, then use the following blend equation and register settings:

Blend Equation (windowA.A is the opacity of window C)

```
       layer2RGB  = (1.0 - windowA.A) * windowB.RGB  +  windowA.A * layer1RGB
```

Register Settings (windowA.A is the opacity of window C)

```
        windowB.srcFactC_Match_Select = NEG_K1_TIMES_DST
        windowB.dstFactC_Match_Select = K1_TIMES_DST
```

If windowA.A is the transparency of window C, then use the following blend equation and register settings:

Blend Equation (windowA.A is the transparency of window C)

```
        layer2RGB  = windowA.A * windowB.RGB  +  (1.0 - windowA.A) * layer1RGB
```

Register Settings (windowA.A is the transparency of window C)

```
        windowB.srcFactC_Match_Select = K1_TIMES_DST
        windowB.dstFactC_Match_Select = NEG_K1_TIMES_DST
```

The above assumes non-premul alpha. Premul alpha would not make sense in this case.

Window C's Alpha is in Window A

```
   layer 4               -----              cursor
   layer 3            ccccccc               windowC no alpha
   layer 2                 bbbbbbbbbb       windowB premul alpha
   layer 1         aaaaaaaaaaaaaaaaaaaaa    windowA alpha for blend w/ c
   layer 0        gggggggggggggggggggggggg  background

   result         gaaaccccccbbbbbbbbaaaag
                        ^   ^ ^
                        |   | |
                        |   | +---- b blended with a
                        |   +------ c blended with (") using alpha from a
                        +---------- c blended with a using alpha from a
```

Same as case 1, but window C contains no alpha values (or garbage alpha values), and the alpha is needed for use from window A to blend window C with window A and window B.

Blend Equations

```
layer0RGBA = BackgroundColorRGBA
layer1RGB  = windowA color
layer1A    = windowA alpha
layer2RGB  = windowB.RGB              +  (1.0 - windowB.A) * layer1RGB
layer2A    = don't care
layer3RGB  = windowA.A * windowC.RGB  +  (1.0 - windowA.A) * layer2RGB
layer3A    = don't care
layer4RGB  = blend with cursor
```

Register Settings

```
    windowA.K1 = 0xff  (use for fade)
    windowA.K2 = 0xff
    windowA.key_Select = NONE
    windowA.srcFactC_Match_Select = K1
    windowA.dstFactC_Match_Select = ZERO
    windowA.srcFactA_Match_Select = K2
    windowA.dstFactA_Match_Select = ZERO

    windowB.K1 = 0xff  (use for fade)
    windowA.K2 = 0xff
```

```
    windowB.key_Select = NONE
    windowB.srcFactC_Match_Select = K1   (K1_TIMES_SRC for non-premul)
    windowB.dstFactC_Match_Select = NEG_K1_TIMES_SRC
    windowB.srcFactA_Match_Select = ZERO
    windowB.dstFactA_Match_Select = K2

    windowC.K1 = 0xff  (use for fade)
    windowC.key_Select = NONE
    windowC.srcFactC_Match_Select = NEG_K1_TIMES_DST
    windowC.dstFactC_Match_Select = K1_TIMES_DST
```

21.3.11 PRISM 3

This section describes the enhancements to PRISM 3. PRISM 3 was formerly known as Smart Dimmer, and so the abbreviation SD is commonly used in register names for PRISM 3.

Features added:

- Programmable statistics window
- Separate crush/adaptation
- Programmable soft clipping
- Increase pixel count limits
- LUT/CSC after enhancement

21.3.11.1 Programmable Statistics Window

This feature allows statistics to be gathered over a subset of the active display. It is useful when part of the display is not interesting for PRISM 3 statistics purposes, such as letterbox or border around a video. Without this feature, the border would influence the histogram and therefore the brightness, giving suboptimal results for the video.

The programming interface is a control bit SD_WINDOW_ENABLE in SD_CONTROL register to enable the feature, and two registers containing position and size of the window. When SD_CONTROL == DISABLE (the default), the position/size registers are ignored, and the entire active display region is used, like in Tegra 3.

The coordinate system is relative to display active area – (0,0) is upper left corner of display active. The active region will have upper left corner (H_POSITION, V_POSITION), and lower right corner (H_POSITION + SD_WIN_H_SIZE – 1, V_POSITION + V_SIZE – 1) inclusive. Note this display coordinate system does not take into account surface rotation or device rotation. It is the same coordinate system use by window position (e.g., B_H_POSITION, B_V_POSITION).

The limitation on minimum pixel count of 2^16 must be obeyed.

The programmable window has no direct effect on enhancement – every pixel of display active area is enhanced.

21.3.11.2 Separate Crush/Adaptation

As in Tegra 3, the AGGRESSIVENESS register controlled both the maximum crushed pixel percentage, and the lowest K (maximum gain). This feature separates them and allows software to separately control lowest K (maximum gain). AGGRESSIVENESS continues to control the maximum crushed pixel percentage.

The programming interface is a control bit K_LIMIT_ENABLE, and a register SD_K_LIMIT.

As in Tegra 3, the initial K (as determined by accumulating histogram bins until maximum crush pixel percentage is reached). When K_LIMIT_ENABLE=ENABLE, the initial K is clamped to SD_K_LIMIT rather than default “lowest_K”. When K_LIMIT_ENABLE = DISABLE (the default), SD_K_LIMIT register is ignored and the default “lowest_L” are used based on AGRESSIVENESS.

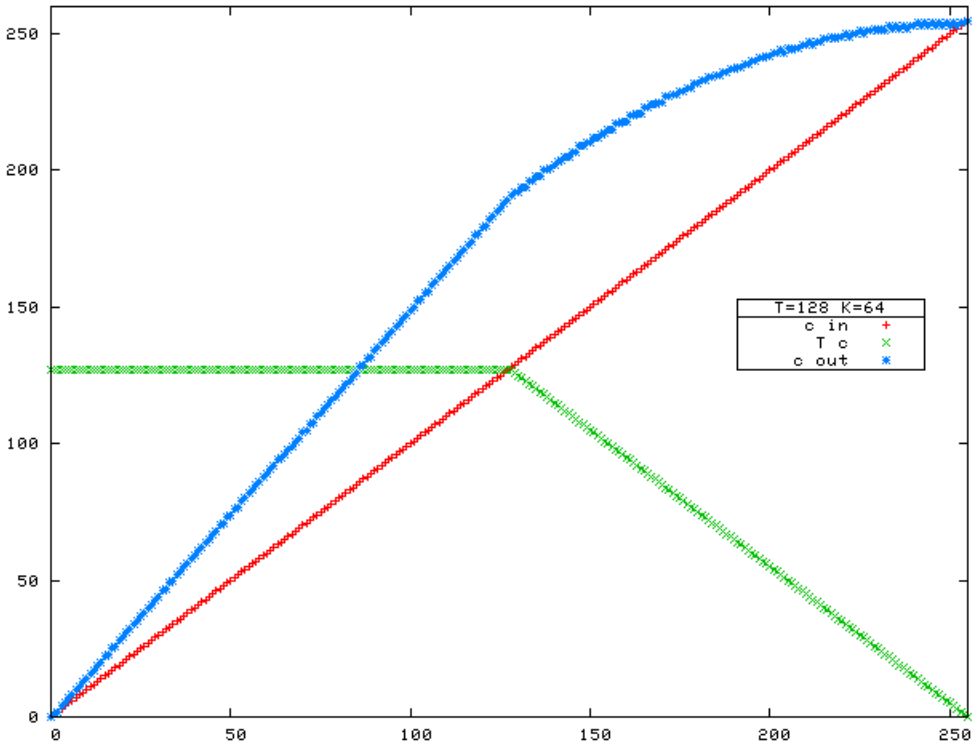
21.3.11.3 Programmable Soft Clipping

In PRISM 3 enhancement, soft clipping is employed to avoid harsh clipping of amplified high-value pixels, and resulting loss of detail in bright areas. The gain applied to high-valued pixels is reduced gradually from 1/K to 1.0, for pixels above a threshold of 128. In Tegra K1 devices, the threshold is fixed at 128, and soft clipping is always enabled. This feature gives more software control.

Since soft clipping is performed after statistics, and it affects the brightness of the image, the statistics and therefore the backlight brightness will be incorrect. This can somewhat compensated for in the backlight transfer-function and/or enhancement value lookup tables.

The Tegra K1 transfer function is shown here, for K=1.5. The T_c plot is the weight applied to fractional portion of K, and goes from 127 to 0, representing 1 to 0.

Figure 62: Default Threshold=128 at K=1.5



With programmable thresholds, other curves are possible. For example, Figure 63 and Figure 64 show more gradual curves using threshold=64 and 0 respectively.

Figure 63: Threshold=64 at K=1.5

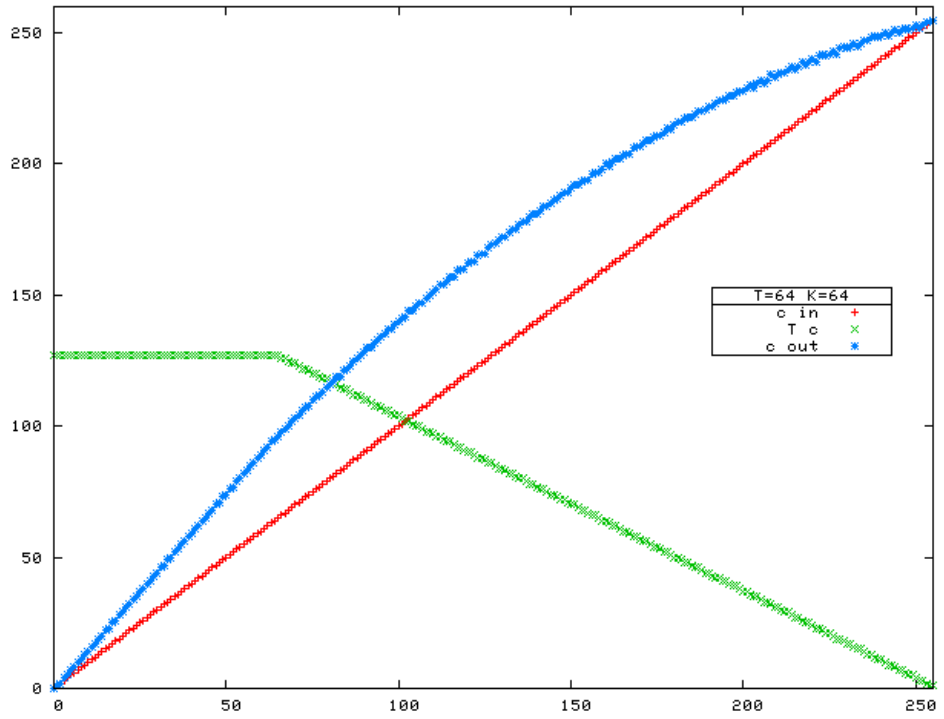
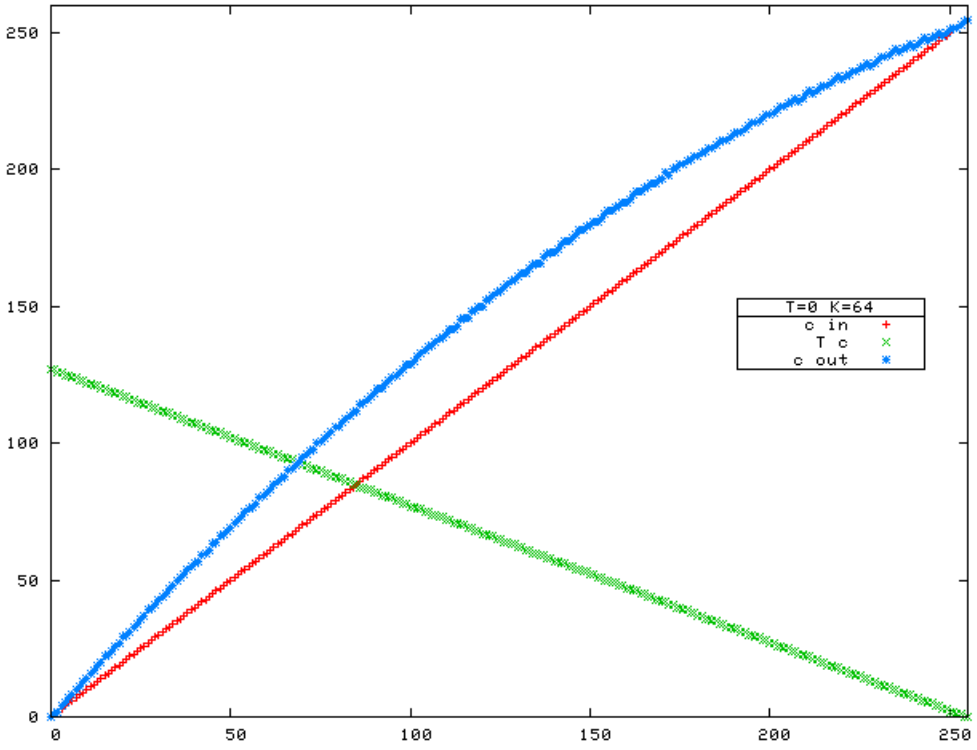


Figure 64: Threshold 0 at K=1.5



Thresholds too high are not usable, because the soft clipping does not kick in early enough to prevent clipping as K grows; note in Figure 65 (threshold 170 at K=1.5) the value of c_out rises about 255. The curve is also not monotonically increasing. Software would need to reduce maximum K to 1.25 or so for such a threshold. Figure 66 shows threshold of 170 with K = 1.25.

Figure 65: Overflowing at Threshold=170 at K=1.5

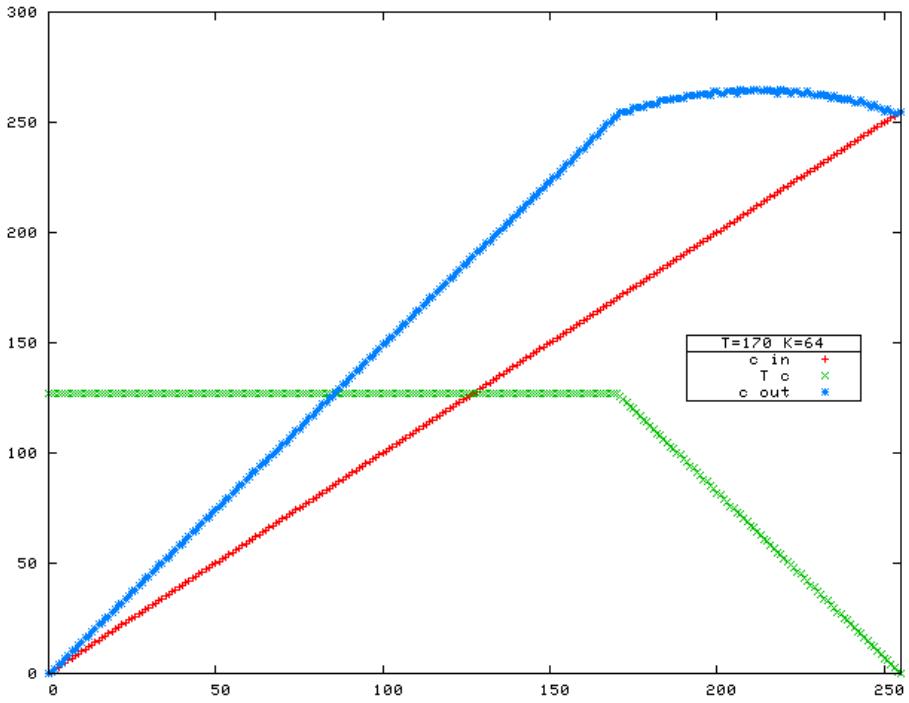
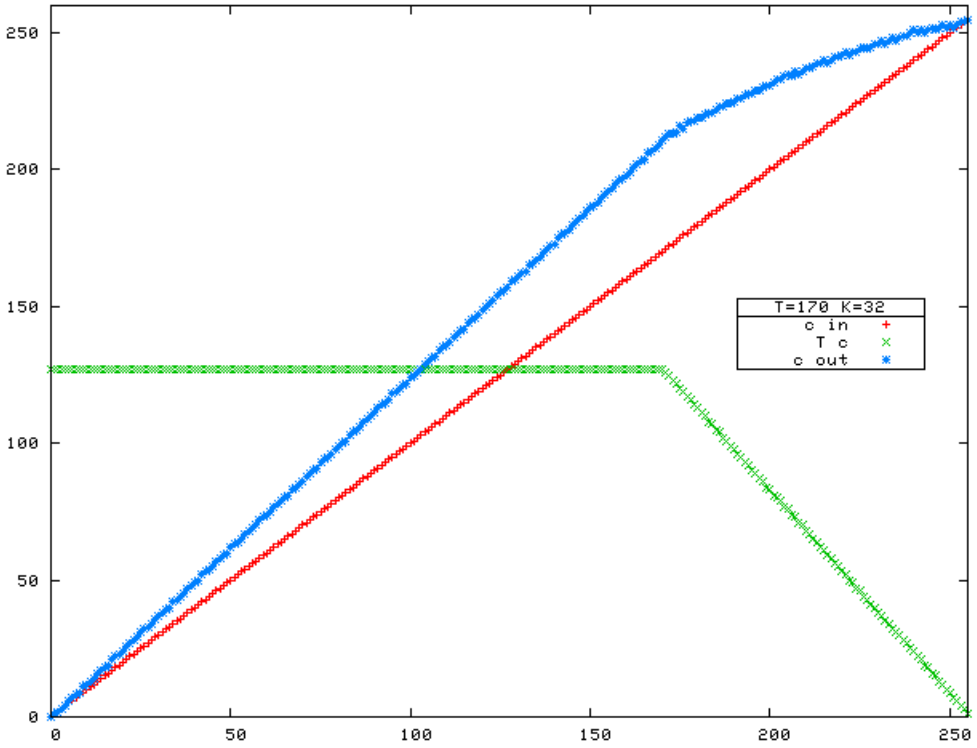


Figure 66: Threshold 170 at K=1.25



The programming interface consists of a control bit SOFT_CLIPPING_ENABLE and one register SD_SOFT_CLIPPING. The latter contains both the threshold and a software-computed reciprocal of inverse threshold. The software-computed reciprocal avoids the need to perform division in hardware. RECIP is computed as 64*1024 * 1.0 / (256 – threshold), truncated to 16-bit unsigned integer. Internally, the hardware may use fewer than 16 bits as described below.

When SOFT_CLIPPING_ENABLE=DISABLE, then SD_SOFT_CLIPPING is ignored and no clipping is done.

When SOFT_CLIPPING_ENABLE=ENABLE, SD_SOFT_CLIPPING is used. At the default values of SD_SOFT_CLIPPING, it behaves like Tegra 3.

The soft clipping equation performed in hardware, per component, is

```
If SD_CONTROL.SOFT_CLIPPING_ENABLE==ENABLE and c_in >= threshold:
        T_c = 0x7f - (((c_in - threshold) * (recip >> 2)) >> 7);
Else
        T_c = 0x7f
```

Where recip is the programmed SOFT_CLIPPING_RECIP; it is shifted right 2 bits to yield a 14-bit value. T_c is a 7-bit weight. The shift by 7 represents discarding 6 bits from recip plus one to give a 7-bit rather than 8-bit result. As before, T_c is used to scale the fractional portion of K:

```
c_k_limited = (K_c * T_c) >> 7;
```

and then apply the fractional gain to the component:

```
c_mul = (c_in * c_k_limited) >> 7;
```

and finally add the integer portion of gain (which is always 1, i.e., 1 * c_in = c_in):

```
c_out = c_in + c_mul
```

21.3.11.4 Increase Pixel Count Limits

Displays are trending to 2560x1600 and larger panels. PRISM 3 needs to support at least 2^{23} pixels. For simplification, the minimum screen size of 2^{16} is retained. Therefore the pixel count dynamic range increases from 32:1 to 128:1. The total_pixels and histogram accumulator counters must support that range. Also, the histogram normalization must support an expanded range of exponents and shifting (0-7 rather than 0-5).

The SD_HISTOGRAM bins need not expand because they contain normalized pixel percentages, not raw counts.

The programming model does not change. SD_PIXEL_COUNT already accommodates this range.

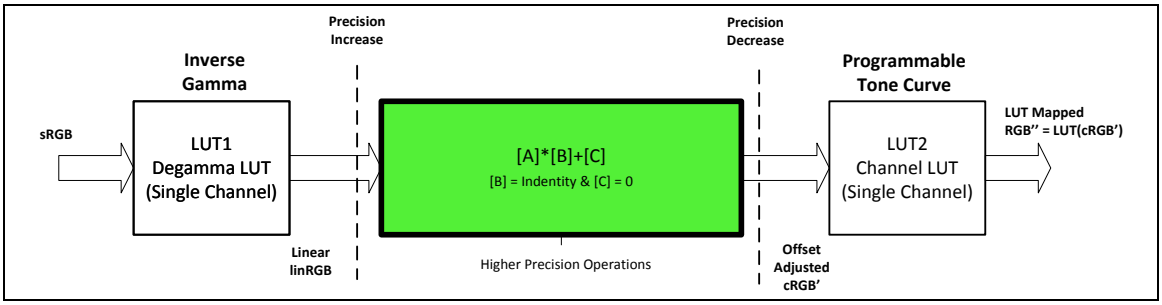
21.3.12 Display Color Management Unit (LUT/CSC)

This unit adds an additional LUT -> CSC -> LUT after the PRISM unit to enable gamma and color gamut change. While this feature is independent from PRISM, and more generally useful, it helps PRISM 3 accommodate non-standard (non-sRGB) panels. It also helps convert from the display's native gamma and gamut (de facto of sRGB) into panels.

- First stage – LUT1 – is an 8-bit to 12-bit RAM-based look-up table (LUT) to convert gamma from sRGB 2.2 to linear (1.0). The output is 8.4 unsigned fixed-point format. A single LUT may be used for all 3 channels.
- Second stage – CSC – is 3x3 matrix multiply, to convert color gamut / whitepoint. 12 bits unsigned in, and 12 bits unsigned out. Its nine coefficients are 10-bit (S1.8 signed).
- Third stage – LUT2 – is a 12-bit to 8-bit LUT to convert from linear gamma to sRGB or similar gamma (for example, 1 / 2.2). A single LUT may be used for all 3 channels.

> **Note:** The window datapaths have subunits called LUT (CP) and CSC already, but they are for different purposes. The CP is for converting linear framebuffer data into sRGB, or for color index modes, or for applying aesthetic changes like contrast. CSC is for converting YCbCr into RGB. They are for normalizing all surfaces into a common format, nominally sRGB, for blending.

Figure 67: Color Management Unit



21.3.12.1 CMU LUT1

LUT1 maps 8-bit gamma-corrected input to 12-bit linear output. A single LUT may be used for all 3 RGB channels – per-channel control is not required.

```
R′ = LUT1[R]
G′ = LUT1[G]
B′ = LUT1[B]
```

21.3.12.2 CSC Matrix

The CSC equation is:

$$\begin{bmatrix} KRR & KGR & KBR \\ KRG & KGG & KBG \\ KRB & KGB & KBB \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

21.3.12.3 CMU LUT2

LUT2 maps 12-bit linear input to 8-bit gamma corrected output. A single LUT may be used for all 3 RGB channels – per-channel control is not required. LUT2 is only 960 rows deep, and divided unevenly so that precision is variable; greater precision (more entries) are allocated to darker values. The first 512 entries 0..511, representing range [0.0, 32.0), are darker values, and are mapped with full 9-bit precision (5.4). The top 448 entries are for lower-precision brighter range [32, 256); they are reduced from 8.4 to 8.1 and offset so that values [32,256) map to 512..959 rather than 576..1023.

```
R’ = LUT2[zone_select(R)]
G’ = LUT2[zone_select(G)]
B’ = LUT2[zone_select(B)]
```

The zone_select function maps a 12-bit component value to 10-bit RAM index, with variable precision:

If C >= THRESHOLD:

C’ = (C >> PRECISION_DIFF) + THRESHOLD – OVERLAP

Else:

C’ = C[9:0]

Where THRESHOLD=2^9 = 512, PRECISION_DIFF = 12 – 9 = 3, and OVERLAP = THRESHOLD >> PRECISION_DIFF = 512 >> 3 = 64.

Software would use an inverse_zone_select to initialize the LUT2.

21.3.12.4 Programming Interface

The CMU LUTs and CSC registers are single-buffered; shadow update and activation have no effect. However, the CMU_ENABLE bit is double-buffered, so it can be enabled/disabled at frame boundaries.

Host interface writes and reads take precedence over the datapath and may result in visible glitches. Software is responsible for ensuring the datapath is idle before writing or reading. An interrupt will be raised upon a conflict. The LUTs are guaranteed to be idle during blanking and when CMU_ENABLE=DISABLE. Software must explicitly enable reading, and reading is only performed for debugging reasons.

Note:	**When reading LUTs:** due to internal RAM latency, after changing read-enable or read address, data is not available until two clocks later. Best way to ensure this is to issue two back-to-back writes of same read address, as shown in read sequence.

Software initialization sequence:

```
Insure CMU_ENABLE=DISABLE or display idle
For C = 0 .. 255:
       C’ = ungamma_function(C/255) * 4095
       CMU_LUT1.LUT1_ADDR = C
       CMU_LUT1.LUT1_DATA = C’
       Write CMU_LUT1
For C = 0 .. 1023: # assume 10-bit
       C’ = gamma_function(inverse_zone_select(C)/4095) * 255
       CMU_LUT2.LUT1_ADDR = C
       CMU_LUT2.LUT1_DATA = C’
       Write CMU_LUT2
Write DISP_COLOR_CONTROL.CMU_ENABLE = ENABLE
Use GENERAL_ACT_REQ
```

Where typically functions are: ungamma_function(x) = $x^{2.2}$, and gamma_function(x) = $x^{1/2.2}$.

LUT read sequence:

```
Insure CMU_ENABLE=DISABLE or display idle
For C = 0 .. 255:
       CMU_LUT1_READ.LUT1_READ_EN = 1
       CMU_LUT1_READ.LUT1_READ_ADDR = C
       Write CMU_LUT1_READ
       Write CMU_LUT1_READ # ensure one cycle delay
       Read CMU_LUT1.LUT1_DATA
Write CMU_LUT1_READ.LUT1_READ_EN = 0
```

21.3.12.5 LUT Conflict Interrupt

To help debug conflicts caused by accessing LUTs while datapath is using them, CMU_LUT_CONFLICT interrupt is raised if:

- host write during scanout (pixel read when cmu_enable=1)
- host read during scanout
- host write during host read (i.e; LUT1_READ_EN=1)

21.3.13 Dither

Temporal dither replaces error diffusion and does not use line buffers.

21.3.14 Output RGB-to-YUV444 CSC

The Tegra K1 display outputs one of three color spaces:

- Red, Green, Blue color primaries.
- YCbCr 4:4:4 Luminance and color difference signals as defined by ITU-R BT.601
- YCbCr 4:4:4 Luminance and color difference signals as defined by ITU-R BT.709

YCbCr color space operates only on limited RGB color space of RGB[16, 235]. HDMI requires YCbCr support of limited color space and optionally extended color space. HDMI also allows for direct RGB limited color space input (that is, RGB[16, 235]). The Tegra K1 Display allows limitation of RGB color space automatically by placing hardcoded full to limited color space scaler unit prior to CSC. This allows for support of the following output formats to HDMI:

- RGB[0, 255]
- RGB[16, 235]
- BT.601 YCbCr 4:4:4 (converted from RGB[16, 235])
- BT.709 YCbCr 4:4:4 (converted from RGB[16, 235])

21.4 Programming Model

21.4.1 Software Alignment Requirements

For Tegra K1 devices, the pixel data alignment restriction is 64B boundaries. The table below describes what registers are affected.

Parameter	Register name	Pitch(packed) 1/2/4 Bpp	Pitch(Planar) 1 Bpp per surface	Block Linear (packed) 1/2/4 Bpp	Block Linear (planar) 1 Bpp per surface
line_stride	B_LINE_STRIDE	64	64	64	64
h_prescale_size	B_H_PRESCALED_SIZE	Bpp	Even	Bpp	Even
v_prescale_size	B_V_PRESCALED_SIZE	NA	Even for 420	NA	Even for 420
h_offset	B_ADDR_H_OFFSET	Bpp	Even	Bpp	Even

Parameter	Register name	Pitch(packed) 1/2/4 Bpp	Pitch(Planar) 1 Bpp per surface	Block Linear (packed) 1/2/4 Bpp	Block Linear (planar) 1 Bpp per surface
v_offset	B_ADDR_V_OFFSET	NA	Even for 420	NA	Even for 420
Startaddr	B_START_ADDR	64	64	512	512

Due to fetch size expanding to 64B, Y lines in planar/semi-planar must be fetched in multiples of 8 for 420, 422, 422R formats.

21.5 Display Controller Registers

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

The display memory map is shown below.

21.5.1 Address Map

Space Name	Offset	Notes
DC_CMD	0x0	Non-shadowed command/sync registers
DC_D_WIN	0x80	Window D instance of DC_WIN
DC_D_WINBUF	0xc0	Window D instance of DC_WINBUF
DC_T_WIN	0x100	Window T instance of DC_WIN
DC_T_WINBUF	0x140	Window T instance of DC_WINBUF
DC_COM	0x300	Non-shadowed non-window registers
DC_DISP	0x400	Shadowed non-window registers
DC_WINC	0x500	Non-shadowed indirect window registers: indirect alias; windows A/B/C; use DISPLAY_WINDOW_HEADER
DC_WIN	0x700	Shadowed indirect window register: indirect alias; windows A/B/C/D; use DISPLAY_WINDOW_HEADER
DC_WINBUF	0x800	Triple-buffered indirect window registers: indirect alias; windows A/B/C/D; use DISPLAY_WINDOW_HEADER
DC_A_WINC	0xa00	Window A instance for DC_WINC
DC_A_WIN	0xb80	Window A instance for DC_WIN
DC_A_WINBUF	0xbc0	Window A instance for DC_WINBUF
DC_B_WINC	0xc00	Window B instance for DC_WINC
DC_B_WIN	0xd80	Window B instance for DC_WIN
DC_B_WINBUF	0xdc0	Window B instance for DC_WINBUF
DC_C_WINC	0xe00	Window C instance for DC_WINC
DC_C_WIN	0xf80	Window C instance for DC_WIN
DC_C_WINBUF	0xfc0	Window C instance for DC_WINBUF
DC_D_WINC	0x1000	Window D instance of DC_WINC (RESERVED)
DC_T_WINC	0x1200	Window T instance of DC_WINC (RESERVED)

21.5.2 New Registers for Tegra K1 Devices

The following registers are new or changed for Tegra K1 devices.

Register	Field	Description
Memfetch registers – Note: Memfetch registers are PER WINDOW		
B_ADDR_H_OFFSET	B_ADDR_H_OFFSET	Horizontal address offset. This is a byte address. It needs to be aligned to Bpp (bytes per pixel). For planar/semi-planar formats, it is the offset of Y surface.
B_ADDR_V_OFFSET	B_ADDR_V_OFFSET	Vertical address offset. For planar/semi-planar surfaces, it is the offset of Y surface. It needs to be even for 420 formats.
B_COLOR_DEPTH	B_COLOR_DEPTH	Supported color formats are described in “Surface/Color Formats”.
B_WIN_OPTIONS	B_H_DIRECTION	Horizontal drawing direction.
	B_V_DIRECTION	Vertical drawing direction.
	B_H_FILTER_ENABLE	Enable for horizontal filter.
	B_V_FILTER_ENABLE	Enable for vertical filter.
	B_V_FILTER_OPTIMIZE	Optimize vertical filter to reduce bandwidth.
	B_V_FILTER_UV_ALIGN	Align UV for vertical filter. 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.
	B_CP_ENABLE	Enable for color palette.
	B_CSC_ENABLE	Enable for color space converter.
	B_DV_ENABLE	Enable for digital vibrance.
	B_YUV_RANGE_EXPAND	Enable range expansion for YUV.
	B_WIN_ENABLE	Window enable.
	B_H_FILTER_MODE	Horizontal filter mode.
	B_SCAN_COLUMN	Load memory columns into display line buffer.
	B_INTERLACE_ENABLE	Interlace Enable

Register	Field	Description
B_H_INITIAL_DDA	B_H_INITIAL_DDA	H initial DDA
B_V_INITIAL_DDA	B_V_INITIAL_DDA	V initial DDA
B_PRESCALED_SIZE	B_H_PRESCALED_SIZE	The H pre-scaled size is needed to determine how many bytes to fetch from memory per line. It is in terms of bytes. It needs to be aligned to bytes per pixel. For planar/semi-planar surfaces, this parameter refers to pre-scale size of Y plane.
	B_V_PRESCALED_SIZE	The V pre-scaled size is needed to determine how many lines to fetch from memory.For planar/semi-planar surfaces, this parameter refers to pre-scale size of Y plane. It needs to be even for 420 formats.
B_LINE_STRIDE	B_LINE_STRIDE	Line stride. It needs to be aligned to 64B
	B_UV_LINE_STRIDE	Line stride for u/v surface
B_START_ADDR	B_START_ADDR	Lower 32 bits of the starting address of the surface. It is aligned to 256B. For planar/semi-planar surfaces, it points to Y surface.
B_START_ADDR_NS	B_START_ADDR_NS	Lower 32 bits of the starting address of the surface. It is aligned to 256B. For planar/semi-planar surfaces, it points to Y surface.
B_START_ADDR_U	B_START_ADDR_U	Lower 32 bits of the starting address of the U surface. It is aligned to 256B. For semi-planar surfaces, it points to UV surface
B_START_ADDR_U_NS	B_START_ADDR_U_NS	Lower 32 bits of the starting address of the U surface. It is aligned to 256B. For semi-planar surfaces, it points to UV surface
B_START_ADDR_V	B_START_ADDR_V	Lower 32 bits of the starting address of the V surface. It is aligned to 256B.
B_START_ADDR_V_NS	B_START_ADDR_V_NS	Lower 32 bits of the starting address of the V surface. It is aligned to 256B.
B_START_ADDR_HI	B_START_ADDR_HI	Higher 32 bits of the starting address of the surface. It is aligned to 256B. For planar/semi-planar surfaces, it points to Y surface.
B_START_ADDR_HI_NS	B_START_ADDR_HI_NS	Higher 32 bits of the starting address of the surface. It is aligned to 256B. For planar/semi-planar surfaces, it points to Y surface.
B_START_ADDR_U_HI	B_START_ADDR_U_HI	Higher 32 bits of the starting address of the U surface. It is aligned to 256B. For semi-planar surfaces, it points to UV surface
B_START_ADDR_U_HI_NS	B_START_ADDR_U_HI_NS	Higher 32 bits of the starting address of the U surface. It is aligned to 256B. For semi-planar surfaces, it points to UV surface

Register	Field	Description
B_START_ADDR_V_HI	B_START_ADDR_V_HI	Higher 32 bits of the starting address of the V surface. It is aligned to 256B.
B_START_ADDR_V_HI_NS	B_START_ADDR_V_HI_NS	Higher 32 bits of the starting address of the V surface. It is aligned to 256B.
B_START_ADDR_FIELD2	B_START_ADDR_FIELD2	Lower 32 bits of the starting address of the surface. It is aligned to 256B. For planar/semi-planar surfaces, it points to Y surface.
B_START_ADDR_FIELD2_NS	B_START_ADDR_FIELD2_NS	Lower 32 bits of the starting address of the surface. It is aligned to 256B. For planar/semi-planar surfaces, it points to Y surface.
B_START_ADDR_U_FIELD2	B_START_ADDR_U_FIELD2	Lower 32 bits of the starting address of the U surface. It is aligned to 256B. For semi-planar surfaces, it points to UV surface
B_START_ADDR_U_FIELD2_NS	B_START_ADDR_U_FIELD2_NS	Lower 32 bits of the starting address of the U surface. It is aligned to 256B. For semi-planar surfaces, it points to UV surface
B_START_ADDR_V_FIELD2	B_START_ADDR_V_FIELD2	Lower 32 bits of the starting address of the V surface. It is aligned to 256B.
B_START_ADDR_V_FIELD2_NS	B_START_ADDR_V_FIELD2_NS	Lower 32 bits of the starting address of the V surface. It is aligned to 256B.
B_START_ADDR_HI_FIELD2	B_START_ADDR_HI_FIELD2	Higher 32 bits of the starting address of the surface. It is aligned to 256B. For planar/semi-planar surfaces, it points to Y surface.
B_START_ADDR_HI_FIELD2_NS	B_START_ADDR_HI_FIELD2_NS	Higher 32 bits of the starting address of the surface. It is aligned to 256B. For planar/semi-planar surfaces, it points to Y surface.
B_START_ADDR_U_HI_FIELD2	B_START_ADDR_U_HI_FIELD2	Higher 32 bits of the starting address of the U surface. It is aligned to 256B. For semi-planar surfaces, it points to UV surface
B_START_ADDR_U_HI_FIELD2_NS	B_START_ADDR_U_HI_FIELD2_NS	Higher 32 bits of the starting address of the U surface. It is aligned to 256B. For semi-planar surfaces, it points to UV surface
B_START_ADDR_V_HI_FIELD2	B_START_ADDR_V_HI_FIELD2	Higher 32 bits of the starting address of the V surface. It is aligned to 256B.
B_START_ADDR_V_HI_FIELD2_NS	B_START_ADDR_V_HI_FIELD2_NS	Higher 32 bits of the starting address of the V surface. It is aligned to 256B.
B_DDA_INCREMENT	B_V_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).
B_SIZE	B_V_SIZE	V size after scaling

Register	Field	Description
B_SURFACE_KIND	B_SURFACE_KIND	Surface kinds supported by Tegra K1 device. enum (PITCH, BL_16B2)
	B_BLOCK_HEIGHT	Height of block in GOBs. enum (HEIGHT_1, HEIGHT_2, HEIGHT_4, HEIGHT_8, HEIGHT_16, HEIGHT_32)
B_SURFACE_WEIGHT	B_SURFACE_WEIGHT_OVERRIDE	Override the default weights of the arbiter in display. 0 – disable, 1 - enable
	B_SURFACE_WEIGHT_Y	Surface weight of Y surface
	B_SURFACE_WEIGHT_U	Surface weight of U surface
	B_SURFACE_WEIGHT_V	Surface weight of V surface
B_UFLOW_STATUS	UFLOW_COUNT	Underflow count. This field indicates the number of contiguous groups of output pixels for which there was no data in the FIFO. For example, if the valid from the FIFO is low for 10 consecutive cycles and then goes high, the counter will increment by one. Reset to zero on write.
	COUNT_OFLOW	Flag bit that indicates that the underflow event counter has overflowed. There were too many events. If COUNT_OFLOW is set, UFLOW_COUNT is meaningless. Cleared on write.
B_LINE_FLIP_DELAY	B_LINE_FLIP_DELAY	Delay (in scan-line-times) between when a hsync happens and when the SYNCPT is incremented for the new LINE_FLIPPED condition.
B_UFLOW_CTRL	B_UFLOW_CTRL_DBG_MODE	When the underflow debug mode is enabled. Instead of sending out the last valid pixel, DBG_PIXEL is sent out.
B_UFLOW_DBG_PIXEL	B_UFLOW_DBG_PIXEL	32-bit debug pixel color.
B_UFLOW_THRESHOLD	B_UFLOW_THRESHOLD	Number of lines to skip when underflow continues to the point where display (XY) is greater than display memfetch request (XY)
B_H_FILTER_HI_P00-F	B_H_FILTER_HI_P0FC0-5	Horizontal Filter phase extended bits.
Color Key registers PER DISPLAY CONTROLLER		
COLOR_KEY0/1_LOWER. R/G/B/A	Key0/1.MinRGBA	RGBA or UYVA lower (32-bpp)
COLOR_KEY0/1_UPPER. R/G/B/A	Key0/1.MaxRGBA	RGBA or UYVA upper (32-bpp)
BACKGROUND_COLOR.R/G/B/A	Background Color.RGBA	RGBA or UYVA (32-bpp)

Register	Field	Description
Cursor registers – Note: Cursor registers are PER DISPLAY CONTROLLER		
CURSOR_START_ADDR	CURSOR_SIZE	Extend enums to: 32x32 64x64 128x128 256x256
Blend registers – Note: Blending registers are PER WINDOW		
Z	BLEND_DEPTH	8-bit depth
K1	BLEND_ALPHA1	8-bit alpha
K2	BLEND_ALPHA2	8-bit alpha
BYPASS	BLEND_BYPASS	boolean
KEY_SELECT	COLOR_KEY_SELECT	3-bit enum: NONE WINDOWA_KEY0 WINDOWA_KEY1 WINDOWB_KEY0 WINDOWB_KEY1 WINDOWC_KEY0 WINDOWC_KEY1
SRCFACTC_MATCH_SELECT	BLEND_MATCH. SRC_FACT_C_SEL	3-bit enum: K1 K1_TIMES_DST NEG_K1_TIMES_DST K1_TIMES_SRC ZERO ONE
DSTFACTC_MATCH_SELECT	BLEND_MATCH. DST_FACT_C_SEL	3-bit enum: K1 K2 K1_TIMES_DST NEG_K1_TIMES_DST NEG_K1_TIMES_SRC NEG_K1 ZERO ONE
SRCFACTA_MATCH_SELECT	BLEND_MATCH. SRC_FACT_A_SEL	3-bit enum: K1 K2 ZERO
DSTFACTA_MATCH_SELECT	BLEND_MATCH. DST_FACT_A_SEL	3-bit enum: K2 ZERO NEG_K1_TIMES_SRC ONE
SRCFACTC_NOMATCH_SELECT	BLEND_NOMATCH. SRC_FACT_C_SEL	3-bit enum: same enums as SRCFACTC_MATCH_SELECT

Register	Field	Description
DSTFACTC_NOMATCH_SELECT	BLEND_NOMATCH. DST_FACT_C_SEL	3-bit enum: same enums as DSTFACTC_MATCH_SELECT
SRCFACTA_NOMATCH_SELECT	BLEND_NOMATCH. SRC_FACT_A_SEL	3-bit enum: same enums as SRCFACTA_MATCH_SELECT
DSTFACTA_NOMATCH_SELECT	BLEND_NOMATCH. DST_FACT_A_SEL	3-bit: same enums as DSTFACTA_MATCH_SELECT
Cursor blend registers – Note: Cursor blend registers are PER DISPLAY CONTROLLER		
K1	BLEND_CURSOR.ALPHA1	8-bit alpha
SRCFACTC_SELECT	BLEND_CURSOR. SRC_FACT_C_SEL	2-bit enum: K1, K1_TIMES_SRC
DSTFACTC_SELECT	BLEND_CURSOR. DST_FACT_C_SEL	2-bit enum: K1, NEG_K1_TIMES_SRC
Dither registers – Note: Dither registers are PER DISPLAY CONTROLLER		
TD_DITHER_PHASE	Temporal Dither phase	2 bits: 0: no frame reset 1,2,3: frame reset to constants 1/2/3
DITHER_CONTROL	Dither enable and method selection	2 bits: 0: DISABLE 2: ORDERED 3: TEMPORAL * (replaces ERRDIFF)
ORD_DITHER_ROTATION	Static/dynamic control	2 bits: 0: STATIC 1/2/3: rotate matrix every N frames

21.5.3 Display Controller Register Definition

Each display supports three windows: Window A, Window B, Window C

The windows may have different features, as described above; however, the register interfaces are the same.

Color Palette

These appear as three instances of 256 32-bit registers with each register consisting of one RGB pixel so not all 32 bits in the registers are used. One instance is used for window A, another instance is used for window B, and the third instance is used for window C. In reality, each instance is implemented as triple 256-word dual-port register files (2P RF) with one read port and one write port (1R1W) and with each word consisting of 1 red/green/blue component. Read port of the color palettes are used for the corresponding window A, window B, or window C and write port of the color palettes are used for host write.

Note that the host cannot read these color palettes, so it must cache this color palette somewhere else to be able to read them. This is done to reduce area.

21.5.4 Display Shadow Registers

Display registers have three shadow types:

1. Not Shadowed

   Writes to these registers take effect immediately.

2. Double Buffered

   Each register has two copies: ASSEMBLY and ACTIVE. ACTIVE is the working copy.

These two copies share the same address/offset.

WRITE_MUX can be programmed to choose which copy the subsequent register writes goes to:

When set to ACTIVE, both ACTIVE and ASSEMBLY copies will be written

When set to ASSEMBLY, only ASSEMBLY copy will be written

READ_MUX can be programmed to choose which copy the subsequent register reads comes from.

If display is in STOP mode, ASSEMBLY copy is latched into ACTIVE copy immediately after

(GENERAL/WIN_A/WIN_B/WIN_C)_ACT_REQ is programmed. In other modes the latching happens on the next frame boundary after (GENERAL/WIN_A/WIN_B/WIN_C)_ACT_REQ is programmed.

3. Triple Buffered

Each register has three copies: ASSEMBLY, ARM, and ACTIVE. ACTIVE is the working copy.

ASSEMBLY and ACTIVE copies share the same address/offset, the ARM copy is located at the address/offset one bigger than the ASSEMBLY/ACTIVE copy.

WRITE_MUX can be programmed to choose which copy the subsequent register writes goes to:

When set to ACTIVE, all three copies will be written

When set to ASSEMBLY, only ASSEMBLY copy will be written

READ_MUX can be programmed to choose which copy the subsequent register reads comes from.

To read back ASSEMBLY or ACTIVE copy, set READ_MUX correctly before register reads.

These two copies share the same address.

To read back ARM copy, do not set READ_MUX, but read back the register/register-field with "_NS" in their names, the offset of which is 1 bigger than its ASSEMBLY/ACTIVE counterpart.

ASSEMBLY copy is latched into ARM copy immediately after GENERAL/WIN_A/WIN_B/WIN_C)_UPDATE is programmed.

If display is in STOP mode, ARM copy is latched into ACTIVE copy immediately after (GENERAL/WIN_A/WIN_B/WIN_C)_ACT_REQ is programmed. In other modes the latching happens on the next frame boundary after (GENERAL/WIN_A/WIN_B/WIN_C)_ACT_REQ is programmed.

21.5.5 Display Control Modes

The DISPLAY_COMMAND is used to set display control mode (**CONTINUOUS**, **ONE-SHOT**, or **STOP**).

Display switches mode when CONTROL_MODE is programmed to shadow register and then activated.

Display enters a mode when the register activation happens, either on frame boundary when entering **STOP** mode, or immediately when exiting **STOP** mode.

In **STOP** mode, display is in idle. Pixel clock is not running.

In **CONTINUOUS** mode, display keeps refreshing the output frame by frame. This mode is mostly used for the panels without internal frame buffer.

In **ONE-SHOT** mode, also known as non-continuous mode, display waits for a trigger before refreshing each frame. Between those frames, display is in idle. This mode is usually for the panels with internal frame buffer.

In ONE-SHOT mode, there are different types of triggers as follows:

- Input triggers

  Input trigger is to notify DISPLAY that a new memory surface is ready to be displayed.

  Host Trigger

  Host trigger is requested when NC_HOST_TRIG_ENABLE is programmed. This register field is in the same register as shadow registers UPDATE/ACT_REQ so that the trigger can happen atomically with register updates

for the new frame. When host trigger is requested within a frame, the requested frame will start immediately after the end of the current frame.

- Output Triggers

  Output trigger is to notify DISPLAY that a panel is ready to receive data from DISPLAY.

  Tear Effect Signal Triggers

  When MSF/SSF register field is set to ENABLE, DISPLAY holds off a frame requested by input triggers until DISPLAY receives a "ready" from the pins that connect to the tearing effect signals from the panel, then sends a new frame to panel. However, if no input trigger has been requested, DISPLAY does not send a new frame.

In CONTINUOUS mode, the meaning of the triggers is different.

- Input Triggers

  Since in CONTINUOUS mode trigger happen automatically and repeatedly, trigger here only affects the buffer address change on a window.

  - Host Trigger

    Host can change the buffer address and then write WIN_A/B/C_ACT_REQ to request the buffer address change. Note that for syncpt logic, it cannot tell if a WIN_A/B/C register activation indicates the change of address, so it always behaves like a change of address happens on any register activation requested by WIN_A/B/C_ACT_REQ. As the result, RD_DONE/OP_DONE is returned.

- Output triggers

  These triggers are not applicable in CONTINUOUS mode. MSF/SSF_ENABLE should never be set in CONTINUOUS mode.

21.5.6 Sync Points

DISPLAY has 4 syncpt clients:

- GENERAL

  Conditions

  - 0: IMMEDIATE return INDX immediately
  - 1: OP_DONE not meaningful, same as IMMEDIATE
  - 2: RD_DONE not meaningful, same as IMMEDIATE
  - 3: REG_WR_SAFE returns INDX whenever it is safe to program GENERAL shadow registers (when all previous GENERAL activation has happened)
  - 4: HSPI returns INDX whenever it is safe to send HSPI data (when all HSPI_*** registers are safe to be programmed)
  - 5: FRAME_DONE returns INDX on the next frame end
  - 6: VPULSE3 returns INDX on the next vpulse3 leading edge (for timing sensitive operations if needed)
  - 7: FRAME_START returns INDX on the next frame start (currently for hardware testing, but can be used if needed)

- WIN_A

  Conditions

  - 0: IMMEDIATE return INDX immediately
  - 1: OP_DONE same as RD_DONE
  - 2: RD_DONE returns INDX when all WIN_A reads for frames activated before the INCR_SYNCPT are complete. In ONE-SHOT mode with TVO disabled, this happens after the last row of window A display. In CONTINUOUS

mode, or in ONE-SHOT mode with TVO enabled, this happens on the last row of window A or on frame end (depending if the INCR_SYNCPT is issued before or after the window A last row).

- 3: REG_WR_SAFE returns INDX whenever it is safe to program WIN_A shadow registers (when all previous WIN_A activation has happened)
- 4: LINE_FLIPPED returns INDX whenver the activation has happened and the previous surface has been completely read out. This is particularly useful for line flip case, when s/w could use the syncpt to free up the previously read surface. As hardware moves on to the new surface.

- 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.

21.5.7 Raise Actions (Legacy)

The Raise action is used to check the state of the display's command execution. The Raise action can be enabled by itself or with other actions.

The raise value is returned to the host controller if the current and all previous commands have completed execution. If the command is executed with the delayed mode, the raise is not returned till that command is executed at the end of the frame.

If there is no pending command when the Raise is issued, the raise value is returned immediately. The raise value is a 4-bit number included in the command.

Context switch acknowledge, register reads, and raises are returned to the host through a read FIFO. However, because there may be more than one raise, read, or acknowledge available for writing to the FIFO in a certain cycle, there must be a priority encoder. Here is the priority:

1. Context switch
2. Register read
3. SIGNAL_RAISE
4. SIGNAL_RAISE1
5. SIGNAL_RAISE2
6. SIGNAL_RAISE3
7. DISP_COMMAND_RAISE
8. HSPI_RAISE
9. Refcount

These packets are for verification; they have no relevance to software. The RAISE packet should include the intended return channel. This channel should be passed back to the host so the RAISE will be reflected in the correct channel.

21.6 Display CMD Registers

21.6.1 DC_CMD_GENERAL_INCR_SYNCPT_0

Offset: 0x0 | Byte Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

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 16 = COND_16 17 = COND_17 18 = COND_18 19 = COND_19 20 = COND_20 21 = COND_21 22 = COND_22 23 = COND_23 24 = COND_24 25 = COND_25 26 = COND_26 27 = COND_27 28 = COND_28 29 = COND_29 30 = COND_30 31 = COND_31
7:0	0x0	GENERAL_INDX: syncpt index value

21.6.2 DC_CMD_GENERAL_INCR_SYNCPT_CNTRL_0

Offset: 0x1 | Byte Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxx0)

Bit	Reset	Description
8	0x0	GENERAL_INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	GENERAL_INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal state of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all Host1x clients affected, then clear all SOFT_RESETs.

21.6.3 DC_CMD_GENERAL_INCR_SYNCPT_ERROR_0

Offset: 0x2 | Byte Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GENERAL_COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

21.6.4 DC_CMD_WIN_A_INCR_SYNCPT_0

Offset: 0x8 | Byte Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

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 16 = COND_16 17 = COND_17 18 = COND_18 19 = COND_19 20 = COND_20 21 = COND_21 22 = COND_22 23 = COND_23 24 = COND_24 25 = COND_25 26 = COND_26 27 = COND_27 28 = COND_28 29 = COND_29 30 = COND_30 31 = COND_31
7:0	0x0	WIN_A_INDX: syncpt index value

21.6.5 DC_CMD_WIN_A_INCR_SYNCPT_CNTRL_0

Offset: 0x9 | Byte Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxx0)

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.

Bit	Reset	Description
0	0x0	WIN_A_INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal states of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all Host1x clients affected, then clear all SOFT_RESETs.

21.6.6 DC_CMD_WIN_A_INCR_SYNCPT_ERROR_0

Offset: 0xa | Byte Offset: 0x28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	WIN_A_COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

21.6.7 DC_CMD_WIN_B_INCR_SYNCPT_0

Offset: 0x10 | Byte Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

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 16 = COND_16 17 = COND_17 18 = COND_18 19 = COND_19 20 = COND_20 21 = COND_21 22 = COND_22 23 = COND_23 24 = COND_24 25 = COND_25 26 = COND_26 27 = COND_27 28 = COND_28 29 = COND_29 30 = COND_30 31 = COND_31
7:0	0x0	WIN_B_INDX: syncpt index value

21.6.8 DC_CMD_WIN_B_INCR_SYNCPT_CNTRL_0

Offset: 0x11 | Byte Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	WIN_B_INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	WIN_B_INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal states of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all Host1x clients affected, then clear all SOFT_RESETs.

21.6.9 DC_CMD_WIN_B_INCR_SYNCPT_ERROR_0

Offset: 0x12 | Byte Offset: 0x48 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	WIN_B_COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

21.6.10 DC_CMD_WIN_C_INCR_SYNCPT_0

Offset: 0x18 | Byte Offset: 0x60 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

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 16 = COND_16 17 = COND_17 18 = COND_18 19 = COND_19 20 = COND_20 21 = COND_21 22 = COND_22 23 = COND_23 24 = COND_24 25 = COND_25 26 = COND_26 27 = COND_27 28 = COND_28 29 = COND_29 30 = COND_30 31 = COND_31
7:0	0x0	WIN_C_INDX: syncpt index value

21.6.11 DC_CMD_WIN_C_INCR_SYNCPT_CNTRL_0

Offset: 0x19 | Byte Offset: 0x64 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	WIN_C_INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	WIN_C_INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal states of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all Host1x clients affected, then clear all SOFT_RESETs.

21.6.12 DC_CMD_WIN_C_INCR_SYNCPT_ERROR_0

Offset: 0x1a | Byte Offset: 0x68 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	WIN_C_COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

21.6.13 DC_CMD_CONT_SYNCPT_VSYNC_0

Offset: 0x28 | Byte Offset: 0xa0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx)

Bit	Reset	Description
8	0x0	VSYNC_EN: On host read bus every time VSYNC (V-blank leading edge) happens and VSYNC_EN is set 0 = DISABLE 1 = ENABLE
7:0	X	VSYNC_INDX: Return INDX (set HOST_CLRD packet TYPE field to SYNCPT)

21.6.14 DC_CMD_CTXSW_0

Context switch registers for class and channel

Should be common to all modules. Includes the current channel/class (writable by software) and the next channel/class (which the hardware sets when it receives a context switch).

Context switch works like this:

Any context switch request triggers an interrupt to the host and causes the new channel/class to be stored in NEXT_CHANNEL/NEXT_CLASS (see vmod/chexample). Software sees that there is a context switch interrupt and does the necessary operations to make the module ready to receive traffic from the new context. It clears the context switch interrupt and writes CURR_CHANNEL/CLASS to the same value as NEXT_CHANNEL/CLASS, which causes a context switch acknowledge packet to be sent to the host. This completes the context switch and allows the host to continue sending data to the module.

Context switches can also be pre-loaded. If CURR_CLASS/CHANNEL are written and updated to the next CLASS/CHANNEL before the context switch request occurs, an acknowledge will be generated by the module and no interrupt will be triggered. This is one way for software to avoid dealing with context switch interrupts.

Another way to avoid context switch interrupts is to set the AUTO_ACK bit.

This bit tells the module to automatically acknowledge any incoming context switch requests without triggering an interrupt. CURR_* and NEXT_* will be updated by the module so they will always be current.

Offset: 0x30 | Byte Offset: 0xc0 | Read/Write: R/W | Reset: 0xf000f800 (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

21.6.15 DC_CMD_DISPLAY_COMMAND_OPTION0_0

Display Controller Option 0. This register is not effective until DISPLAY_COMMAND is written.

Class: Display Command

Offset: 0x31 | Byte Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx000xxxxxxx00000000)

Bit	Reset	Description
18	0x0	WINDOW_C_NC_DISPLAY: Window C Non-Continuous Display. This is effective only in Non-Continuous Display mode when window B buffer switching is not controlled by host. If this bit is enabled, a frame is sent whenever Window B buffer is switched. 0 = DISABLE 1 = ENABLE
17	0x0	WINDOW_B_NC_DISPLAY: Window B Non-Continuous Display. This is effective only in Non-Continuous Display mode when window B buffer switching is not controlled by host. If this bit is enabled, a frame is sent whenever Window B buffer is switched. 0 = DISABLE 1 = ENABLE
16	0x0	WINDOW_A_NC_DISPLAY: Window A Non-Continuous Display. This is effective only in Non-Continuous Display mode when window A buffer switching is not controlled by host. If this bit is enabled, a frame is sent whenever Window A buffer is switched. 0 = DISABLE 1 = ENABLE
7:6	0x0	SSF_SOURCE: Source pin for the SSF input. Controls which pin will be used as the source for the trigger input when MSF mode is enabled. Note that although the same pins are available for both MSF and SSF, the order in the enum and hence the values differ between the pins. This is to maintain backwards compatibility with previous chips, which had a fixed mapping. The init value and the first value in the enum reflect this historical mapping. 0= LCD_DC pin (legacy default) 1= LCD_SPI pin 2= LCD_SDI pin 3= RESERVED for future use. 0 = SSF_LDC 1 = SSF_LSPI 2 = SSF_LSDI
5	0x0	SSF_ENABLE: Sub-Display Stop Frame (SSF) input. This is effective only in Non-Continuous Display mode. When enabled, SSF signal can be input through LDC pin. When SSF is enabled a trigger to send a frame in Non-Continuous Display mode will be delayed until SSF is active. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	0x0	SSF_POLARITY: Sub-Display Stop Frame (SSF) Polarity 0= Active high 1= Active low
3:2	0x0	MSF_SOURCE: Source pin for the MSF input. Controls which pin will be used as the source for the trigger input when MSF mode is enabled. Note that although the same pins are available for both MSF and SSF, the order in the enum and hence the values differ between the pins. This is to maintain backwards compatibility with previous chips, which had a fixed mapping. The init value and the first value in the enum reflects this historical mapping. 0= LCD_SPI pin (legacy default) 1= LCD_DC pin 2= LCD_SDI pin 3= RESERVED for future use. 0 = MSF_LSPI 1 = MSF_LDC 2 = MSF_LSDI
1	0x0	MSF_ENABLE: Main-Display Stop Frame (MSF) input This is effective only in Non-Continuous Display mode. When enabled, the MSF signal can be input through the LSPI pin. When MSF is enabled, a trigger to send a frame in Non-Continuous Display mode will be delayed until MSF is active. 0 = DISABLE 1 = ENABLE
0	0x0	MSF_POLARITY: Main-Display Stop Frame (MSF) Polarity 0= Active high 1= Active low

21.6.16 DC_CMD_DISPLAY_COMMAND_0

Display Command

Offset: 0x32 | Byte Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00xxxx0)

Bit	Reset	Description
30:27	X	DISP_COMMAND_RAISE_CHANNEL_ID: Display Command Channel ID
26:22	X	DISP_COMMAND_RAISE_VECTOR: Display Command Raise Vector. This raise vector is at the next line or frame boundary, depending on GENERAL_ACT_CNTR_SEL
6:5	0x0	DISPLAY_CTRL_MODE: Display Controller Mode 0= Stop Display, this can be used to stop sending frame at the next frame boundary. This is automatically generated in Non-Continuous Display after sending one frame. If this is issued when display controller is already stopped then there is no frame sent. Raise vector (if raise is enabled) is also returned immediately. This command can also be used in non-continuous display mode to stop accepting non-host trigger conditions from other clients. 1= Continuous Display, the display controller will continuously send frame. Continuous display mode can be stopped by switching to Non-Continuous Display or by issuing Stop Display. 2= Non-Continuous Display, the display controller is forced to send one frame of each active display and then wait for the next time this command is issued or for other (non-host) trigger conditions to send frame. The sending of frames may be delayed by MSF or SSF input signals from the display device. If a Stop Display is issued while in non-continuous display mode then non-host trigger conditions will no longer be accepted until the next time Non-Continuous Display is issued 0 = STOP 1 = C_DISPLAY 2 = NC_DISPLAY
0	0x0	DISP_COMMAND_RAISE: Display Command Raise. Raise vector will be returned at the end of command completion 0 = DISABLE 1 = ENABLE

21.6.17 DC_CMD_SIGNAL_RAISE_0

When written, the next occurrence of the selected SIGNAL will cause a RAISE to be sent to the host. Software must not write this register if a previous request (from previous write) is still outstanding. Added SIGNAL_RAISE_TYPE option so that multiple raises can be returned without software intervention. SIGNAL_RAISE1, SIGNAL_RAISE2, or SIGNAL_RAISE3 can be used if more than one source signal is needed.

Offset: 0x33 | Byte Offset: 0xcc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0xxxxxxxxxxx)

Bit	Reset	Description
19:16	X	SIGNAL_RAISE_CHANNEL_ID: Signal Raise Channel ID
12	0x0	SIGNAL_RAISE_TYPE: 0= One-shot, single raise returned. 1= Continuous, raise is returned persistently whenever raise event is true until this register is reprogrammed such that SIGNAL_RAISE_SELECT=NONE or SIGNAL_RAISE_TYPE=ONESHOT 0 = ONESHOT 1 = CONT
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

21.6.18 DC_CMD_DISPLAY_POWER_CONTROL_0

PW0, PW1, PW2, and PW3 are tied internally to the same value corresponding to PW3.

Software is required to program all these register fields to ENABLE - PW0 to PW3 simultaneously, in the same register write access.

The power sequencing provided by PW0 - PW3 was LCD output specific, which is not applicable to Tegra K1 devices.

Display Power Control

Offset: 0x36 | Byte Offset: 0xd8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00xxxxx0x0xxxxxxx0x0x0x0x0)

Bit	Reset	Description
25	0x0	HSPI_ENABLE: Host SPI write cycle enable. SPI_ENABLE must be enabled also for this bit to be effective. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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
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

21.6.19 DC_CMD_INT_STATUS_0

This reflects status of all pending interrupts which is valid as long as the interrupt is not cleared even if the interrupt is masked. A pending interrupt can be cleared by writing a '1' to this the corresponding interrupt status bit in this register.

Display Interrupt and Status

Offset: 0x37 | Byte Offset: 0xdc | Read/Write: R/W | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25	X	WIN_T_UF_INT: Window T Underflow Interrupt Status 0= interrupt not pending 1= interrupt pending
24	X	WIN_D_UF_INT: Window D Underflow Interrupt Status 0= interrupt not pending 1= interrupt pending
23	X	HC_UF_INT: Cursor Underflow Interrupt Status 0= interrupt not pending 1= interrupt pending

Bit	Reset	Description
22	X	CMU_LUT_CONFLICT_INT: CMU LUT read/write conflict; write 1 to clear
16	X	WIN_C_OF_INT: Window C Overflow Interrupt Status 0= interrupt not pending 1= interrupt pending
15	X	WIN_B_OF_INT: Window B Overflow Interrupt Status 0= interrupt not pending 1= interrupt pending
14	X	WIN_A_OF_INT: Window A Overflow Interrupt Status 0= interrupt not pending 1= interrupt pending
13	X	SSF_INT: Sub-Display Stop Frame Interrupt Status 0= interrupt not pending 1= interrupt pending
12	X	MSF_INT: Main-Display Stop Frame Interrupt Status 0= interrupt not pending 1= interrupt pending
10	X	WIN_C_UF_INT: Window C Underflow Interrupt Status 0= interrupt not pending 1= interrupt pending
9	X	WIN_B_UF_INT: Window B Underflow Interrupt Status 0= interrupt not pending 1= interrupt pending
8	X	WIN_A_UF_INT: Window A Underflow Interrupt Status 0= interrupt not pending 1= interrupt pending
7	X	SPI_BUSY_INT: SPI Busy Interrupt Status 0= interrupt not pending 1= interrupt pending
5	X	V_PULSE2_INT: Vertical Pulse 2 Interrupt 0= interrupt not pending 1= interrupt pending
4	X	V_PULSE3_INT: Vertical Pulse 3 Interrupt 0= interrupt not pending 1= interrupt pending
3	X	H_BLANK_INT: Horizontal Blank Interrupt 0= interrupt not pending 1= interrupt pending
2	X	V_BLANK_INT: Vertical Blank Interrupt 0= interrupt not pending 1= interrupt pending
1	X	FRAME_END_INT: Frame End Interrupt 0= interrupt not pending 1= interrupt pending

Bit	Reset	Description
0	X	CTXSW_INT: Context Switch Interrupt Status (this is cleared on write) 0= interrupt not pending 1= interrupt pending

21.6.20 DC_CMD_INT_MASK_0

Setting bits in this register masks the corresponding interrupt but neither clears a pending interrupt nor prevents a pending interrupt from being generated. Masking an interrupt also does not clear a pending interrupt status and does not prevent a pending interrupt status from being generated.

Interrupt Mask

Offset: 0x38 | Byte Offset: 0xe0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx0000xxxxx00000x0000x000000)

Bit	Reset	Description
25	0x0	WIN_T_UF_INT_MASK: Window T Underflow Interrupt Mask. 0 = MASKED 1 = NOTMASKED
24	0x0	WIN_D_UF_INT_MASK: Window D Underflow Interrupt Mask. 0 = MASKED 1 = NOTMASKED
23	0x0	HC_UF_INT_MASK: Cursor Underflow Interrupt Mask. 0 = MASKED 1 = NOTMASKED
22	0x0	CMU_LUT_CONFLICT_INT_MASK: CMU LUT read/write conflict. 0 = MASKED 1 = NOTMASKED
16	0x0	WIN_C_OF_INT_MASK: Window C Overflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
15	0x0	WIN_B_OF_INT_MASK: Window B Overflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
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

Bit	Reset	Description
8	0x0	WIN_A_UF_INT_MASK: Window A Underflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
7	0x0	SPI_BUSY_INT_MASK: SPI Busy Interrupt Mask 0 = MASKED 1 = NOTMASKED
5	0x0	V_PULSE2_INT_MASK: Vertical Pulse 2 Interrupt Mask 0 = MASKED 1 = NOTMASKED
4	0x0	V_PULSE3_INT_MASK: Vertical Pulse 3 Interrupt Mask 0 = MASKED 1 = NOTMASKED
3	0x0	H_BLANK_INT_MASK: Horizontal Blank Interrupt Mask 0 = MASKED 1 = NOTMASKED
2	0x0	V_BLANK_INT_MASK: Vertical Blank Interrupt Mask 0 = MASKED 1 = NOTMASKED
1	0x0	FRAME_END_INT_MASK: Frame End Interrupt Mask 0 = MASKED 1 = NOTMASKED
0	0x0	CTXSW_INT_MASK: Context Switch Interrupt Mask 0 = MASKED 1 = NOTMASKED

21.6.21 DC_CMD_INT_ENABLE_0

Setting bits in this register enable the corresponding interrupt event to generate a pending interrupt. Interrupt output signal will be activated only if the corresponding interrupt is not masked.

Disabling an interrupt will not clear a corresponding pending interrupt - it only prevents a new interrupt event to generate a pending interrupt.

Interrupt Enable

Offset: 0x39 | Byte Offset: 0xe4 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxx0000xxxxx00000x0000x000001)

Bit	Reset	Description
25	0x0	WIN_T_UF_INT_ENABLE: Window T Underflow Interrupt Enable 0 = DISABLE 1 = ENABLE
24	0x0	WIN_D_UF_INT_ENABLE: Window D Underflow Interrupt Enable 0 = DISABLE 1 = ENABLE
23	0x0	HC_UF_INT_ENABLE: Cursor Underflow Interrupt Enable 0 = DISABLE 1 = ENABLE
22	0x0	CMU_LUT_CONFLICT_INT_ENABLE: CMU LUT read/write conflict 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
16	0x0	WIN_C_OF_INT_ENABLE: Window C Overflow Interrupt Enable 0 = DISABLE 1 = ENABLE
15	0x0	WIN_B_OF_INT_ENABLE: Window B Overflow Interrupt Enable 0 = DISABLE 1 = ENABLE
14	0x0	WIN_A_OF_INT_ENABLE: Window A Overflow Interrupt Enable 0 = DISABLE 1 = ENABLE
13	0x0	SSF_INT_ENABLE: Sub-Display Stop Frame Interrupt Enable 0 = DISABLE 1 = ENABLE
12	0x0	MSF_INT_ENABLE: Main-Display Stop Frame Interrupt Enable 0 = DISABLE 1 = ENABLE
10	0x0	WIN_C_UF_INT_ENABLE: Window C Underflow Interrupt Enable 0 = DISABLE 1 = ENABLE
9	0x0	WIN_B_UF_INT_ENABLE: Window B Underflow Interrupt Enable 0 = DISABLE 1 = ENABLE
8	0x0	WIN_A_UF_INT_ENABLE: Window A Underflow Interrupt Enable 0 = DISABLE 1 = ENABLE
7	0x0	SPI_BUSY_INT_ENABLE: SPI Busy Interrupt Enable 0 = DISABLE 1 = ENABLE
5	0x0	V_PULSE2_INT_ENABLE: Vertical Pulse 2 Interrupt Enable 0 = DISABLE 1 = ENABLE
4	0x0	V_PULSE3_INT_ENABLE: Vertical Pulse 3 Interrupt Enable 0 = DISABLE 1 = ENABLE
3	0x0	H_BLANK_INT_ENABLE: Horizontal Blank Interrupt Enable 0 = DISABLE 1 = ENABLE
2	0x0	V_BLANK_INT_ENABLE: Vertical Blank Interrupt Enable 0 = DISABLE 1 = ENABLE
1	0x0	FRAME_END_INT_ENABLE: Frame End Interrupt Enable 0 = DISABLE 1 = ENABLE
0	0x1	CTXSW_INT_ENABLE: Context Switch Interrupt Enable 0 = DISABLE 1 = ENABLE

21.6.22 DC_CMD_INT_TYPE_0

Two interrupt types are available:

- Edge interrupt - transition on input signal/event generates pending interrupt
- Level interrupt - active level on input signal/event generates pending interrupt

Interrupt Type

Offset: 0x3a | Byte Offset: 0xe8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx0000xxxxx00000x0000x00000x)

Bit	Reset	Description
25	0x0	WIN_T_UF_INT_TYPE: Window T Underflow Interrupt Type 0 = EDGE 1 = LEVEL
24	0x0	WIN_D_UF_INT_TYPE: Window D Underflow Interrupt Type 0 = EDGE 1 = LEVEL
23	0x0	HC_UF_INT_TYPE: Cursor Underflow Interrupt Type 0 = EDGE 1 = LEVEL
22	0x0	CMU_LUT_CONFLICT_INT_TYPE: CMU LUT read/write conflict. Must be EDGE. 0 = EDGE 1 = LEVEL
16	0x0	WIN_C_OF_INT_TYPE: Window C Overflow Interrupt Type 0 = EDGE 1 = LEVEL
15	0x0	WIN_B_OF_INT_TYPE: Window B Overflow Interrupt Type 0 = EDGE 1 = LEVEL
14	0x0	WIN_A_OF_INT_TYPE: Window A Overflow Interrupt Type 0 = EDGE 1 = LEVEL
13	0x0	SSF_INT_TYPE: Sub-Display Stop Frame Interrupt Type 0 = EDGE 1 = LEVEL
12	0x0	MSF_INT_TYPE: Main-Display Stop Frame Interrupt Type 0 = EDGE 1 = LEVEL
10	0x0	WIN_C_UF_INT_TYPE: Window C Underflow Interrupt Type 0 = EDGE 1 = LEVEL
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

Bit	Reset	Description
5	0x0	V_PULSE2_INT_TYPE: Vertical Pulse 2 Interrupt Type 0 = EDGE 1 = LEVEL
4	0x0	V_PULSE3_INT_TYPE: Vertical Pulse 3 Interrupt Type 0 = EDGE 1 = LEVEL
3	0x0	H_BLANK_INT_TYPE: Horizontal Blank Interrupt Type 0 = EDGE 1 = LEVEL
2	0x0	V_BLANK_INT_TYPE: Vertical Blank Interrupt Type 0 = EDGE 1 = LEVEL
1	0x0	FRAME_END_INT_TYPE: Frame End Interrupt Type 0 = EDGE 1 = LEVEL

21.6.23 DC_CMD_INT_POLARITY_0

For edge interrupts, these bits specify whether a pending interrupt is generated on the falling edge or on the rising edge of the corresponding input signal/event. For level interrupts, these bits specify whether a pending interrupt is generated on the low level or on the high level of the corresponding input signal/event. 0 rw CTXSW_INT_POLARITY init=0 // Context Switch Interrupt Polarity enum (LOW, HIGH) // 0= falling edge or low level interrupt // 1= rising edge or high level interrupt.

Interrupt Polarity

Offset: 0x3b | Byte Offset: 0xec | Read/Write: R/W | Reset: 0x00400000 (0bxxxxxx0001xxxxx00000x0000x00000x)

Bit	Reset	Description
25	0x0	WIN_T_UF_INT_POLARITY: Window T Underflow Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
24	0x0	WIN_D_UF_INT_POLARITY: Window D Underflow Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
23	0x0	HC_UF_INT_POLARITY: Cursor Underflow Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
22	HIGH	CMU_LUT_CONFLICT_INT_POLARITY: CMU LUT read/write conflict. Must be HIGH. 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH

Bit	Reset	Description
16	0x0	WIN_C_OF_INT_POLARITY: Window C Overflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
15	0x0	WIN_B_OF_INT_POLARITY: Window B Overflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
14	0x0	WIN_A_OF_INT_POLARITY: Window A Overflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
13	0x0	SSF_INT_POLARITY: Sub-Display Stop Frame. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
12	0x0	MSF_INT_POLARITY: Main-Display Stop Frame. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
10	0x0	WIN_C_UF_INT_POLARITY: Window C Underflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
9	0x0	WIN_B_UF_INT_POLARITY: Window B Underflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
8	0x0	WIN_A_UF_INT_POLARITY: Window A Underflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
7	0x0	SPI_BUSY_INT_POLARITY: SPI Busy. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH

Bit	Reset	Description
5	0x0	V_PULSE2_INT_POLARITY: V Pulse 2 Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
4	0x0	V_PULSE3_INT_POLARITY: V Pulse 3. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
3	0x0	H_BLANK_INT_POLARITY: H Blank. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
2	0x0	V_BLANK_INT_POLARITY: V Blank. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
1	0x0	FRAME_END_INT_POLARITY: Frame End. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH

21.6.24 DC_CMD_SIGNAL_RAISE1_0

When written, next occurrence of the selected SIGNAL will cause a RAISE to be sent to the host. Software must not write this register if a previous request (from previous write) is still outstanding. Added SIGNAL_RAISE1_TYPE option so that multiple raises can be returned without software intervention.

Offset: 0x3c | Byte Offset: 0xf0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0xxxxxxxxxxx)

Bit	Reset	Description
19:16	X	SIGNAL_RAISE1_CHANNEL_ID: Signal Raise Channel ID
12	0x0	SIGNAL_RAISE1_TYPE: 0= One-shot, single raise returned. 1= Continuous, raise is returned persistently whenever raise event is true until this register is reprogrammed such that SIGNAL_RAISE1_SELECT=NONE or SIGNAL_RAISE1_TYPE=ONESHOT 0 = ONESHOT 1 = CONT

Bit	Reset	Description
10:8	X	SIGNAL_RAISE1_SELECT: which signal to raise on. 0= none, no raise sent back 1= Frame End signal 2= V Blank signal 3= V Pulse 3 signal 4= Rising edge of V Blank signal 5= Falling edge of V Blank signal 6= Rising edge of V Pulse 3 signal 7= Falling edge of V Pulse 3 signal 0 = NONE 1 = FRAME_END 2 = VBLANK 3 = VPULSE3 4 = VBLANK_START 5 = VBLANK_END 6 = VPULSE3_START 7 = VPULSE3_END
4:0	X	SIGNAL_RAISE1_VECTOR: bit number to raise

21.6.25 DC_CMD_SIGNAL_RAISE2_0

When written, next occurrence of the selected SIGNAL will cause a RAISE to be sent to the host. Software must not write this register if a previous request (from previous write) is still outstanding. Added SIGNAL_RAISE2_TYPE option so that multiple raises can be returned without software intervention.

Offset: 0x3d | Byte Offset: 0xf4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0xxxxxxxxxxx)

Bit	Reset	Description
19:16	X	SIGNAL_RAISE2_CHANNEL_ID: Signal Raise Channel ID
12	0x0	SIGNAL_RAISE2_TYPE: 0= One-shot, single raise returned 1= Continuous, raise is returned persistently whenever raise event is true until this register is reprogrammed such that SIGNAL_RAISE2_SELECT=NONE or SIGNAL_RAISE2_TYPE=ONESHOT 0 = ONESHOT 1 = CONT
10:8	X	SIGNAL_RAISE2_SELECT: which signal to raise on 0= none, no raise sent back 1= Frame End signal 2= V Blank signal 3= V Pulse 3 signal 4= Rising edge of V Blank signal 5= Falling edge of V Blank signal 6= Rising edge of V Pulse 3 signal 7= Falling edge of V Pulse 3 signal 0 = NONE 1 = FRAME_END 2 = VBLANK 3 = VPULSE3 4 = VBLANK_START 5 = VBLANK_END 6 = VPULSE3_START 7 = VPULSE3_END
4:0	X	SIGNAL_RAISE2_VECTOR: bit number to raise

21.6.26 DC_CMD_SIGNAL_RAISE3_0

When written, next occurrence of the selected SIGNAL will cause a RAISE to be sent to the host. Software must not write this register if a previous request (from previous write) is still outstanding. Added SIGNAL_RAISE3_TYPE option so that multiple raises can be returned without software intervention.

Offset: 0x3e | Byte Offset: 0xf8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0xxxxxxxxxxx)

Bit	Reset	Description
19:16	X	SIGNAL_RAISE3_CHANNEL_ID: Signal Raise Channel ID
12	0x0	SIGNAL_RAISE3_TYPE: 0= One-shot, single raise returned. 1= Continuous, raise is returned persistently whenever raise event is true until this register is reprogrammed such that SIGNAL_RAISE3_SELECT=NONE or SIGNAL_RAISE3_TYPE=ONESHOT 0 = ONESHOT 1 = CONT
10:8	X	SIGNAL_RAISE3_SELECT: which signal to raise on 0= none, no raise sent back 1= Frame End signal 2= V Blank signal 3= V Pulse 3 signal 4= Rising edge of V Blank signal 5= Falling edge of V Blank signal 6= Rising edge of V Pulse 3 signal 7= Falling edge of V Pulse 3 signal 0 = NONE 1 = FRAME_END 2 = VBLANK 3 = VPULSE3 4 = VBLANK_START 5 = VBLANK_END 6 = VPULSE3_START 7 = VPULSE3_END
4:0	X	SIGNAL_RAISE3_VECTOR: bit number to raise

21.6.27 DC_CMD_STATE_ACCESS_0

Double/triple buffers read and write access control

Offset: 0x40 | Byte Offset: 0x100 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0x0)

Bit	Reset	Description
2	0x0	WRITE_MUX: Write access control 0= write assembly state 1= write active state When set to ACTIVE, register writes also propagate to assembly set for double buffered registers, to both assembly and arm set for triple buffered registers. 0 = ASSEMBLY 1 = ACTIVE

Bit	Reset	Description
0	0x0	READ_MUX: Read access control 0= read assembly state 1= read active state ARM state register read is not controlled by this mux, but by reading the registers with "_NS" suffix 0 = ASSEMBLY 1 = ACTIVE

21.6.28 DC_CMD_STATE_CONTROL_0

State Control for Activating/Arming New Register State

Restrictions: ACT_REQ cannot be programmed at the same time the corresponding "UPDATE" is programmed.

If so desired, it should be split into two consecutive writes to this register.

Offset: 0x41 | Byte Offset: 0x104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx0xxxxxxx0xx000000xx00000)

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
15	0x0	CURSOR_UPDATE: Trigger for the 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
12	0x0	WIN_D_UPDATE: Trigger for the arming state (from assembly to armed state) for the win D subset of the triple buffered registers. This register is also known as "Update Register" 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
7	0x0	CURSOR_ACT_REQ: Non-window specific 0= no request pending/request completed 1= activation requested/pending 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	0x0	WIN_D_ACT_REQ: Window D activation request 0= no request pending/request completed 1= activation requested/pending 0 = DISABLE 1 = ENABLE
3	0x0	WIN_C_ACT_REQ: Window C activation request 0= no request pending/request completed. 1= activation requested/pending 0 = DISABLE 1 = ENABLE
2	0x0	WIN_B_ACT_REQ: Window B activation request 0= no request pending/request completed. 1= activation requested/pending 0 = DISABLE 1 = ENABLE
1	0x0	WIN_A_ACT_REQ: Window A activation request 0= no request pending/request completed. 1= activation requested/pending 0 = DISABLE 1 = ENABLE
0	0x0	GENERAL_ACT_REQ: Non-window-specific 0= no request pending/request completed. 1= activation requested/pending 0 = DISABLE 1 = ENABLE

21.6.29 DC_CMD_DISPLAY_WINDOW_HEADER_0

Display Window Header for programming display windows and their corresponding buffer start addresses.

Class: Display Window Programming Header

Offset: 0x42 | Byte Offset: 0x108 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000xxxx)

Bit	Reset	Description
7	0x0	WINDOW_D_SELECT: Window D Select 0= disable window D programming 1= enable window D programming 0 = DISABLE 1 = ENABLE
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

Bit	Reset	Description
4	0x0	WINDOW_A_SELECT: Window A Select. 0= disable window A programming. 1= enable window A programming 0 = DISABLE 1 = ENABLE

21.6.30 DC_CMD_REG_ACT_CONTROL_0

Register activation options

Offset: 0x43 | Byte Offset: 0x10c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx0xx00x0x0x0)

Bit	Reset	Description
10	0x0	WIN_D_ACT_CNTR_SEL: Select which counter to use for window D activation 0= vertical counter 1= horizontal counter 0 = VCOUNTER 1 = HCOUNTER
7	0x0	CURSOR_ACT_CNTR_SEL: Select which counter to use for window C activation 0= vertical counter 1= horizontal counter 0 = VCOUNTER 1 = HCOUNTER
6	0x0	WIN_C_ACT_CNTR_SEL: Select which counter to use for window C activation 0= vertical counter 1= horizontal counter 0 = VCOUNTER 1 = HCOUNTER
4	0x0	WIN_B_ACT_CNTR_SEL: Select which counter to use for window B activation 0= vertical counter 1= horizontal counter 0 = VCOUNTER 1 = HCOUNTER
2	0x0	WIN_A_ACT_CNTR_SEL: Select which counter to use for window A activation 0= vertical counter 1= horizontal counter 0 = VCOUNTER 1 = HCOUNTER
0	0x0	GENERAL_ACT_CNTR_SEL: Select which counter to use for general activation 0= vertical counter 1= horizontal counter 0 = VCOUNTER 1 = HCOUNTER

21.6.31 DC_CMD_WIN_T_STATE_CONTROL_0

WIN_T State control register 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.
- When SECURE_CONTROL.SECURE_ENABLE=1, this register is only accessible when HOST1X secure bit is set.
- If SECURE_CONTROL.SECURE_ENABLE is zero, this register is accessible regardless of HOST1X secure bit level.

Offset: 0x44 | Byte Offset: 0x110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	WIN_T_UPDATE: Trigger for the 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
0	0x0	WIN_T_ACT_REQ: 0= no request pending/request completed 0 = DISABLE 1 = ENABLE

21.6.32 DC_CMD_SECURE_CONTROL_0

Secure control register for enabling secure mode and controlling host1x register loads and reads

Restrictions:

- This register is only accessible when the HOST1X secure bit is set. If the HOST1X secure bit is zero writes to this register are ignored and reads return 0.

Offset: 0x45 | Byte Offset: 0x114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxx000000xxxxx000)

Bit	Reset	Description
13	DISABLE	SECURE_SOR_PROTECT: if ENABLE, blanks the display output to SOR when SECURE_ENABLE is true. 0 = DISABLE 1 = ENABLE
12	DISABLE	SECURE_DSIB_PROTECT: if ENABLE, blanks the display output to DSIB when SECURE_ENABLE is true. 0 = DISABLE 1 = ENABLE
11	DISABLE	SECURE_DSIA_PROTECT: If ENABLE, blanks the display output to DSIA when SECURE_ENABLE is true. 0 = DISABLE 1 = ENABLE
10	DISABLE	SECURE_HDMI_PROTECT: If ENABLE, blanks the display output to HDMI when SECURE_ENABLE is true. 0 = DISABLE 1 = ENABLE
9	ASSEMBLY	SECURE_READ_MUX: Controls which of the double-buffered window registers are read in TZ Secure mode (host1x secure bit set). Similar to STATE_ACCESS.READ_MUX 0 = ASSEMBLY 1 = ACTIVE
8	ASSEMBLY	SECURE_WRITE_MUX: Controls which of the double-buffered window registers are written in TZ Secure mode (host1x secure bit set). Similar to STATE_ACCESS.WRITE_MUX 0 = ASSEMBLY 1 = ACTIVE
2	DISABLE	SECURE_CRC_PROTECT: If ENABLE, and SECURE_ENABLE=ENABLE, CRC computation is disabled. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	DISABLE	SECURE_CMU_PROTECT: If ENABLE, and SECURE_ENABLE=ENABLE, CMU registers are only accessible in TZ Secure mode . 0 = DISABLE 1 = ENABLE
0	DISABLE	SECURE_ENABLE: Double-buffered: Active state changed at next frame boundary, or immediately in STOP mode. Read returns active state. Changes window T to and from Secure mode 0 = DISABLE 1 = ENABLE

21.6.33 DC_CMD_WIN_D_INCR_SYNCPT_0

Offset: 0x4c | Byte Offset: 0x130 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	WIN_D_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 16 = COND_16 17 = COND_17 18 = COND_18 19 = COND_19 20 = COND_20 21 = COND_21 22 = COND_22 23 = COND_23 24 = COND_24 25 = COND_25 26 = COND_26 27 = COND_27 28 = COND_28 29 = COND_29 30 = COND_30 31 = COND_31
7:0	0x0	WIN_D_INDX: syncpt index value

21.6.34 DC_CMD_WIN_D_INCR_SYNCPT_CNTRL_0

Offset: 0x4d | Byte Offset: 0x134 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	WIN_D_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.

Bit	Reset	Description
0	0x0	WIN_D_INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal states of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all Host1x clients affected, then clear all SOFT_RESETs.

21.6.35 DC_CMD_WIN_D_INCR_SYNCPT_ERROR_0

Offset: 0x4e | Byte Offset: 0x138 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	WIN_D_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.

21.7 Display COM Registers

21.7.1 DC_COM_CRC_CONTROL_0

CRC Control

CRC is provided for at speed testing and diagnostic. When CRC is enabled, the CRC logic waits for the next VSync pulse or the one after that (depending on CRC_WAIT) and then it captures one frame of data at the end of display pipeline and computes the CRC value.

After one frame of data is captured, the CRC logic will stop capturing data.

When CRC_INPUT_DATA = FULL_FRAME, DISPLAY_COMMAND.DISPLAY_CTRL_MODE should be programmed to C_DISPLAY so that CRC works properly.

When CRC_INPUT_DATA = ACTIVE_DATA, it can work on both NC_DISPLAY and C_DISPLAY modes, and can work for multiple frames if CRC is checked, disabled, and re-enabled after the end of frame v-active area and before next VSYNC.

CRC logic takes 8 bits of control signals and 24-bit RGB pixel after dither and after display color (R and B) swap option.

Input [31:0] into CRC depends on CRC_INPUT_DATA. If programmed as FULL_FRAME, input data is { LVP1, LPV0, LHP2, LHP1, LHP0, VSYNC, HSYNC, ACTIVE, R[7:0], G[7:0], B[7:0]} and CRC runs over the entire frame (including blank). If programmed as ACTIVE_DATA, input data is {R[7:0], G[7:0], B[7:0]} and CRC runs only during active display area.

Offset: 0x300 | Byte Offset: 0xc00 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	0x0	CRC_ALWAYS: CRC always: calculates CRC for every following frame. Must use with CRC_INPUT_DATA = ACTIVE_DATA. If enabled, CRC_WAIT field is ignored. 0 = DISABLE 1 = ENABLE
2	0x0	CRC_INPUT_DATA: CRC input data 0= Full frame (RGB data and control) 1= Active display (Only RGB data) 0 = FULL_FRAME 1 = ACTIVE_DATA
1	0x0	CRC_WAIT: CRC Wait 0= 1 Vsync 1= 2 Vsyncs

Bit	Reset	Description
0	0x0	CRC_ENABLE: CRC Enable 0 = DISABLE 1 = ENABLE

21.7.2 DC_COM_CRC_CHECKSUM_0

CRC Checksum

This register can be read by host after CRC logic stops capturing data.

Offset: 0x301 | Byte Offset: 0xc04 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	0x0	CRC_CHECKSUM: CRC Checksum

21.7.3 DC_COM_PIN_MISC_CONTROL_0

Pin Miscellaneous Control

Pin Miscellaneous Control

Offset: 0x31b | Byte Offset: 0xc6c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0xx)

Bit	Reset	Description
2	0x0	DISP_CLOCK_OUTPUT: Display Clock (DCLK) Enable. 0= disable. 1= enable display clock to be output on LCD_DE pin (LCD_DE output select must be appropriately programmed for this to be effective) 0 = DISABLE 1 = ENABLE

21.7.4 DC_COM_PM0_CONTROL_0

PM0 Signal Control

Class: Pulse Width Modulation

PM0 signal is programmable pulse width modulation signal that can be output on several pins. The control register should be initialized once before PM0 is enabled.

Note: The period is expressed as multiples of 4 times the divider value.
The actual period - in clock cycles - is given by:
period = (1 + PM0_CLOCK_DIVIDER) * ((PM0_PERIOD + 1) * 4)

Offset: 0x31c | Byte Offset: 0xc70 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:18	X	PM0_PERIOD: PM0 Period (4, 8, ... , 256 clock cycles)
17:4	X	PM0_CLOCK_DIVIDER: PM0 Clock Divider (1 to 16384)

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 are running only when the PW0 signal is enabled. 2) In non-continuous mode, shift clock and pixel clock run continuously, but line clock and frame clock only run while a frame is being sent.

21.7.5 DC_COM_PM0_DUTY_CYCLE_0

PM0 Duty Cycle

A counter repeatedly counts up from 0 to ((PM0_PERIOD << 2) + 3) pre-scaled cycles.

The period always starts with the output value == 1. After DUTY_CYCLE number of pre-scaled cycles, the output value is cleared for the remainder of the period. In order to output constant 0, simply set the DUTY_CYCLE to 0. To output a constant 1, set DUTY_CYCLE to be any value greater than ((PM0_PERIOD << 2) + 3).

Offset: 0x31d | Byte Offset: 0xc74 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8:0	X	PM0_DUTY_CYCLE: PM0 Duty Cycle (or D) From 1/P to D/P pulse high time where P is the period.

21.7.6 DC_COM_SCRATCH_REGISTER_A_0

Scratch Register A

Class: Software Scratch Registers

Offset: 0x325 | Byte Offset: 0xc94 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH_REGISTER_A: Scratch Register A

21.7.7 DC_COM_SCRATCH_REGISTER_B_0

Scratch Register B

Offset: 0x326 | Byte Offset: 0xc98 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH_REGISTER_B: Scratch Register B

21.7.8 DC_COM_CRC_CHECKSUM_LATCHED_0

CRC Checksum latched

This register is a latched version of CRC_CHECKSUM. Latching happens at frame end.

> **Note:** CRC_INPUT_DATA needs to be set to ACTIVE_DATA if this register is used. In full frame mode, CRC is frozen two cycles after frame end due to pipelining, so only in active area mode, CRC is consistent and independent of display control mode, and can be checked continuously frame by frame.

Offset: 0x329 | Byte Offset: 0xca4 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	0x0	CRC_CHECKSUM_LATCHED: CRC Checksum latched

21.7.9 DC_COM_CMU_CSC_KRR_0

CMU Color Space Conversion Matrix

- R' = sat(KRR * R + KGR * G + KBR * B)
- G' = sat(KRG * R + KGG * G + KBG * B)
- B' = sat(KRB * R + KGB * G + KBB * B)

Coefficients are signed 1.8 fixed point, for a range of approximately [-2.0, +1.996]

Offset: 0x32a | Byte Offset: 0xca8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KRR

21.7.10 DC_COM_CMU_CSC_KGR_0

Offset: 0x32b | Byte Offset: 0xcac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KGR

21.7.11 DC_COM_CMU_CSC_KBR_0

Offset: 0x32c | Byte Offset: 0xcb0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KBR

21.7.12 DC_COM_CMU_CSC_KRG_0

Offset: 0x32d | Byte Offset: 0xcb4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KRG

21.7.13 DC_COM_CMU_CSC_KGG_0

Offset: 0x32e | Byte Offset: 0xcb8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KGG

21.7.14 DC_COM_CMU_CSC_KBG_0

Offset: 0x32f | Byte Offset: 0xcbc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KBG

21.7.15 DC_COM_CMU_CSC_KRB_0

Offset: 0x330 | Byte Offset: 0xcc0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KRB

21.7.16 DC_COM_CMU_CSC_KGB_0

Offset: 0x331 | Byte Offset: 0xcc4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KGB

21.7.17 DC_COM_CMU_CSC_KBB_0

Offset: 0x332 | Byte Offset: 0xcc8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KBB

21.7.18 DC_COM_CMU_LUT_MASK_0

CMU LUT Mask

LUT channel data write enables

Offset: 0x333 | Byte Offset: 0xccc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx00xxxxxx00)

Bit	Reset	Description
9:8	0x0	LUT2_WR_MASK: Color channel write mask 0 = enable all channels 1-3 = enable red/green/blue only 0 = ALL 1 = RED 2 = GREEN 3 = BLUE
1:0	0x0	LUT1_WR_MASK: Color channel write mask 0 = enable all channels 1-3 = enable red/green/blue only 0 = ALL 1 = RED

Bit	Reset	Description
		2 = GREEN 3 = BLUE

21.7.19 DC_COM_CMU_LUT1_0

Entries are written indirectly.

Ensure CMU_ENABLE=DISABLE or display idle

For C = 0 .. 255:

C' = ungamma_function(C/255) * ((1 << NV_DISPLAY_CMU_DP_WIDTH) - 1)

Write CMU_LUT1.LUT1_ADDR = C, LUT1_DATA = C'

Write DISP_COLOR_CONTROL.CMU_ENABLE = ENABLE

Use GENERAL_ACT_REQ to activate CMU_ENABLE

CMU LUT1

Offset: 0x336 | Byte Offset: 0xcd8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
27:16	X	LUT1_DATA: Readable only when LUT1_READ_EN=1
7:0	0x0	LUT1_ADDR

21.7.20 DC_COM_CMU_LUT2_0

Write sequence:

Ensure CMU_ENABLE=DISABLE or display idle

For C = 0 .. NV_DISPLAY_CMU_LUT2_DEPTH-1:

C' = gamma_function(inverse_zone_select(C)/((1 << NV_DISPLAY_CMU_DP_WIDTH) - 1)) * 255

Write CMU_LUT2.LUT2_ADDR = C, LUT2_DATA = C'

Write DISP_COLOR_CONTROL.CMU_ENABLE = ENABLE

Use GENERAL_ACT_REQ

CMU LUT2

Offset: 0x337 | Byte Offset: 0xcdc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
23:16	X	LUT2_DATA: Readable only when LUT2_READ_EN=1
9:0	0x0	LUT2_ADDR

21.8 Display DISP Registers

These registers control DISP display only; not including the DISP display window parameters.

21.8.1 DC_DISP_DISP_SIGNAL_OPTIONS0_0

Display Signal Options 0

Offset: 0x400 | Byte Offset: 0x1000 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx0x0xxx000x0xxx0x0x0xxxxxxxx)

Bit	Reset	Description
26	0x0	M1_ENABLE: M1 Enable 0 = DISABLE 1 = ENABLE
24	0x0	M0_ENABLE: M0 Enable 0 = DISABLE 1 = ENABLE
20	0x0	V_PULSE3_ENABLE: V Pulse 3 Enable 0 = DISABLE 1 = ENABLE
19	0x0	V_PULSE2_ENABLE: V Pulse 2 Enable 0 = DISABLE 1 = ENABLE
18	0x0	V_PULSE1_ENABLE: V Pulse 1 Enable 0 = DISABLE 1 = ENABLE
16	0x0	V_PULSE0_ENABLE: V Pulse 0 Enable 0 = DISABLE 1 = ENABLE
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

21.8.2 DC_DISP_DISP_WIN_OPTIONS_0

Display Window Options

Offset: 0x402 | Byte Offset: 0x1008 | Read/Write: R/W | Reset: 0x00000000 (0bx00xxx0xxxxxxxx0xxxxxxxxxxxxxxxx)

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

Bit	Reset	Description
25	0x0	SOR_ENABLE: SOR interface - LVDS/eDP The SOR unit must also be separately enabled in its own register space in order to use HDMI functionality. 0 = DISABLE 1 = ENABLE
16	0x0	CURSOR_ENABLE: Cursor Enable 0 = DISABLE 1 = ENABLE

21.8.3 DC_DISP_DISP_TIMING_OPTIONS_0

Display Timing Options

Class: Display Standard Timings

Programming of display timing registers must meet these restrictions:

- Constraint 1: H_REF_TO_SYNC + H_SYNC_WIDTH + H_BACK_PORCH > 20.
- Constraint 2: V_REF_TO_SYNC + V_SYNC_WIDTH + V_BACK_PORCH > 1.
- Constraint 3: V_FRONT_PORCH + V_SYNC_WIDTH + V_BACK_PORCH > 1 (vertical blank).
- Constraint 4: V_SYNC_WIDTH >= 1

  H_SYNC_WIDTH >= 1
- Constraint 5: V_REF_TO_SYNC >= 1

  H_REF_TO_SYNC >= 0
- Constraint 6: V_FRONT_PORCH >= (V_REF_TO_SYNC + 1)

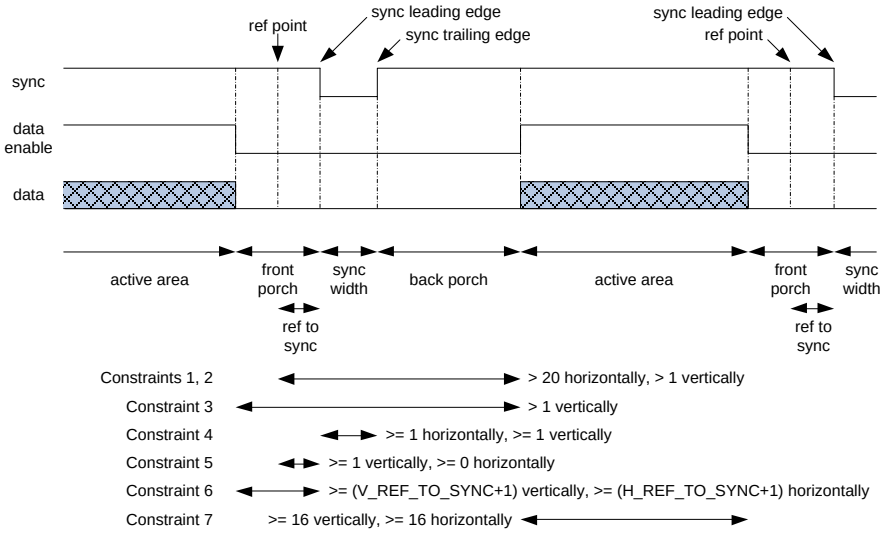
  H_FRONT_PORCH >= (H_REF_TO_SYNC + 1)
- Constraint 7: H_DISP_ACTIVE >= 16

  V_DISP_ACTIVE >= 16

Timing Diagram

- This diagram applies to both vertical and horizontal timing
- The back porch is the only parameter that can be negative

Figure 68: Display Timing Options Timing Diagram



This register specifies display timing options for HSYNC and VSYNC.

Offset: 0x405 | Byte Offset: 0x1014 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000000000000)

Bit	Reset	Description
12:0	0x0	VSYNC_H_POSITION: VSYNC Horizontal Position. This parameter specifies the position where VSYNC can toggle with respect to H reference point.

21.8.4 DC_DISP_REF_TO_SYNC_0

H/V Reference to Sync

This register specifies the start position of HSYNC and VSYNC with respect to H and V reference point (line and frame start) correspondingly. The H and V reference points correspond to the time when H and V display timing counter is re-initialized to zero correspondingly.

The H reference point also determines the point where V display timing counter is incremented so this points the horizontal relationship between HSYNC and VSYNC.

Note: VSYNC's rising/falling edge is fixed at H reference point zero.

Offset: 0x406 | Byte Offset: 0x1018 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0000000000000xxx0000000000000)

Bit	Reset	Description
28:16	0x0	V_REF_TO_SYNC: V reference to VSYNC (minimum 1 line clock)
12:0	0x0	H_REF_TO_SYNC: H reference to HSYNC (minimum 0 pixel clock)

21.8.5 DC_DISP_SYNC_WIDTH_0

H/V SYNC Pulse Width

This register specifies the width of HSYNC and VSYNC pulses. Check the Display Timing Options section for programming restrictions for REF_TO_SYNC.

Offset: 0x407 | Byte Offset: 0x101c | Read/Write: R/W | Reset: 0x00000000 (0bxxx0000000000000xxx0000000000000)

Bit	Reset	Description
28:16	0x0	V_SYNC_WIDTH: VSYNC pulse width (minimum 1 line clock)
12:0	0x0	H_SYNC_WIDTH: HSYNC pulse width (minimum 1 pixel clock)

21.8.6 DC_DISP_BACK_PORCH_0

H/V Back Porch

This register specifies the distance between H/V SYNC trailing edge to beginning of display active area. This is a 2's complement value, and a negative value indicates that H/V SYNC overlaps with the corresponding display active area. Check the Display Timing Options section for programming restrictions for REF_TO_SYNC.

Offset: 0x408 | Byte Offset: 0x1020 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0000000000000xxx0000000000000)

Bit	Reset	Description
28:16	0x0	V_BACK_PORCH: V back porch
12:0	0x0	H_BACK_PORCH: H back porch

21.8.7 DC_DISP_DISP_ACTIVE_0

H/V Display Active Width

This register specifies the width of the H/V display active area. Check the Display Timing Options section for programming restrictions for REF_TO_SYNC.

Offset: 0x409 | Byte Offset: 0x1024 | Read/Write: R/W | Reset: 0x00030003 (0bxxx0000000000011xxx0000000000011)

Bit	Reset	Description
28:16	0x3	V_DISP_ACTIVE: V display active width (minimum 16 lines)
12:0	0x3	H_DISP_ACTIVE: H display active width (minimum 16 pixels)

21.8.8 DC_DISP_FRONT_PORCH_0

H/V Front Porch

This register specifies the distance between end of H/V display active area to the leading edge of the corresponding H/V SYNC.

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 Display Timing Options section for programming restrictions for REF_TO_SYNC.

Offset: 0x40a | Byte Offset: 0x1028 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0000000000000xxx0000000000000)

Bit	Reset	Description
28:16	0x0	V_FRONT_PORCH: VSYNC front porch (minimum -=PS_=-V_REF_TO_SYNC + 1)
12:0	0x0	H_FRONT_PORCH: HSYNC front porch (minimum -=PS_=-H_REF_TO_SYNC + 1)

21.8.9 DC_DISP_H_PULSE0_CONTROL_0

H Pulse 0 Control

Class: Display Extended Timings

Horizontal pulse 0 is programmable pulse that repeats every line.

In NORMAL mode, this signal can have several pulses (A to D) per line with programmable width as defined by the pairs of start and end positions. The pulses must not overlap and must occur in sequence: pulse A, then pulse B, etc. In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

In ONE_CLOCK mode, this signal can have up to twice the number of pulses per line with each pulse having a width of 1 pixel clock. In this mode, the position of the one-clock pulses corresponds to the enabled Start and End positions.

Regardless of the mode, the pulse generator processes the pairs of start and end position sequentially in the order of: Start A, End A, Start B, End B, etc.

So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

Polarity adjustment is made prior to V display qualification. This register specifies options for Horizontal pulse 0.

Offset: 0x40b │ Byte Offset: 0x102c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	H_PULSE0_LAST: H Pulse 0 Last point. 0= end on Start A position 1= end on End A position 2= end on Start B position 3= end on End B position 4= end on Start C position 5= end on End C position 6= end on Start D position 7= end on End D position others= reserved 0 = START_A 1 = END_A 2 = START_B 3 = END_B 4 = START_C 5 = END_C 6 = START_D 7 = END_D
7:6	X	H_PULSE0_V_QUAL: H Pulse 0 Vertical Qualifier. 0= always running 2= run during vertical active area 3= run during vertical active plus 1 line 0 = ALWAYS 2 = VACTIVE 3 = VACTIVE1
4	X	H_PULSE0_POLARITY: H Pulse 0 Polarity. Polarity adjustment is done before the vertical qualifier is applied. 0 = HIGH 1 = LOW

Bit	Reset	Description
3	X	H_PULSE0_MODE: H Pulse 0 Mode 0= Normal mode 1= Single-clock mode 0 = NORMAL 1 = ONE_CLOCK

21.8.10 DC_DISP_H_PULSE0_POSITION_A_0

H Pulse 0 Position A

Offset: 0x40c │ Byte Offset: 0x1030 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE0_END_A: H Pulse 0 End A (minimum -=PS_=-H_PULSE0_START_A+1)
12:0	X	H_PULSE0_START_A: H Pulse 0 Start A (minimum 0)

21.8.11 DC_DISP_H_PULSE0_POSITION_B_0

H Pulse 0 Position B

Offset: 0x40d │ Byte Offset: 0x1034 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE0_END_B: H Pulse 0 End B (minimum -=PS_=-H_PULSE0_START_B+1)
12:0	X	H_PULSE0_START_B: H Pulse 0 Start B (minimum -=PS_=-H_PULSE0_END_A+1)

21.8.12 DC_DISP_H_PULSE0_POSITION_C_0

H Pulse 0 Position C

Offset: 0x40e │ Byte Offset: 0x1038 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE0_END_C: H Pulse 0 End C (minimum -=PS_=-H_PULSE0_START_C+1)
12:0	X	H_PULSE0_START_C: H Pulse 0 Start C (minimum -=PS_=-H_PULSE0_END_B+1)

21.8.13 DC_DISP_H_PULSE0_POSITION_D_0

H Pulse 0 Position D

Offset: 0x40f │ Byte Offset: 0x103c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE0_END_D: H Pulse 0 End D (minimum -=PS_=-H_PULSE0_START_D+1)
12:0	X	H_PULSE0_START_D: H Pulse 0 Start D (minimum -=PS_=-H_PULSE0_END_C+1)

21.8.14 DC_DISP_H_PULSE1_CONTROL_0

H Pulse 1 Control

Horizontal pulse 1 is programmable pulse that repeats every line.

In NORMAL mode, this signal can have several pulses (A to D) per line with programmable width as defined by the pairs of start and end positions. The pulses must not overlap and must occur in sequence: pulse A, then pulse B, etc. In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

In ONE_CLOCK mode, this signal can have up to twice the number of pulses per line with each pulse having a width of 1 pixel clock. In this mode, the position of the one-clock pulses corresponds to the enabled Start and End positions.

Regardless of the mode, the pulse generator processes the pairs of start and end position sequentially in the order of: Start A, End A, Start B, End B, etc.

So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value), then the pulse generator will stop at the last valid position.

Polarity adjustment is made prior to V display qualification. This register specifies options for Horizontal pulse 1.

Offset: 0x410 │ Byte Offset: 0x1040 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	H_PULSE1_LAST: H Pulse 1 Last point 0= end on Start A position 1= end on End A position 2= end on Start B position 3= end on End B position 4= end on Start C position 5= end on End C position 6= end on Start D position 7= end on End D position others= reserved 0 = START_A 1 = END_A 2 = START_B 3 = END_B 4 = START_C 5 = END_C 6 = START_D 7 = END_D
7:6	X	H_PULSE1_V_QUAL: H Pulse 1 Vertical Qualifier 0= always running 2= run during vertical active area 3= run during vertical active plus 1 line 0 = ALWAYS 2 = VACTIVE 3 = VACTIVE1
4	X	H_PULSE1_POLARITY: H Pulse 1 Polarity. Polarity adjustment is done before the vertical qualifier is applied 0 = HIGH 1 = LOW
3	X	H_PULSE1_MODE: H Pulse 1 Mode 0= Normal mode 1= Single-clock mode 0 = NORMAL 1 = ONE_CLOCK

21.8.15 DC_DISP_H_PULSE1_POSITION_A_0

H Pulse 1 Position A

Offset: 0x411 | Byte Offset: 0x1044 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE1_END_A: H Pulse 1 End A (minimum -=PS_=-H_PULSE1_START_A+1)
12:0	X	H_PULSE1_START_A: H Pulse 1 Start A (minimum 0)

21.8.16 DC_DISP_H_PULSE1_POSITION_B_0

H Pulse 1 Position B

Offset: 0x412 | Byte Offset: 0x1048 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE1_END_B: H Pulse 1 End B (minimum -=PS_=-H_PULSE1_START_B+1)
12:0	X	H_PULSE1_START_B: H Pulse 1 Start B (minimum -=PS_=-H_PULSE1_END_A+1)

21.8.17 DC_DISP_H_PULSE1_POSITION_C_0

H Pulse 1 Position C

Offset: 0x413 | Byte Offset: 0x104c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE1_END_C: H Pulse 1 End C (minimum -=PS_=-H_PULSE1_START_C+1)
12:0	X	H_PULSE1_START_C: H Pulse 1 Start C (minimum -=PS_=-H_PULSE1_END_B+1)

21.8.18 DC_DISP_H_PULSE1_POSITION_D_0

H Pulse 1 Position D

Offset: 0x414 | Byte Offset: 0x1050 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE1_END_D: H Pulse 1 End D (minimum -=PS_=-H_PULSE1_START_D+1)
12:0	X	H_PULSE1_START_D: H Pulse 1 Start D (minimum -=PS_=-H_PULSE1_END_C+1)

21.8.19 DC_DISP_H_PULSE2_CONTROL_0

H Pulse 2 Control

Horizontal pulse 2 is a programmable pulse that repeats every line.

In NORMAL mode, this signal can have several pulses (A to D) per line with programmable width as defined by the pairs of start and end positions. The pulses must not overlap and must occur in sequence: pulse A, then pulse B, etc. In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse

generator will stop as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

In ONE_CLOCK mode this signal can have up to twice the number of pulses per line with each pulse having a width of 1 pixel clock. In this mode, the position of the one-clock pulses corresponds to the enabled Start and End positions.

Regardless of the mode, the pulse generator processes the pairs of start and end position sequentially in the order of: Start A, End A, Start B, End B, etc. So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value), then the pulse generator will stop at the last valid position. Polarity adjustment is made prior to V display qualification. This register specifies options for Horizontal pulse 2.

Offset: 0x415 │ Byte Offset: 0x1054 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	H_PULSE2_LAST: H Pulse 2 Last point 0= end on Start A position 1= end on End A position 2= end on Start B position 3= end on End B position 4= end on Start C position 5= end on End C position 6= end on Start D position 7= end on End D position others= reserved 0 = START_A 1 = END_A 2 = START_B 3 = END_B 4 = START_C 5 = END_C 6 = START_D 7 = END_D
7:6	X	H_PULSE2_V_QUAL: H Pulse 2 Vertical Qualifier 0= always running 2= run during vertical active area 3= run during vertical active plus 1 line 0 = ALWAYS 2 = VACTIVE 3 = VACTIVE1
4	X	H_PULSE2_POLARITY: H Pulse 2 Polarity. Polarity adjustment is done before the vertical qualifier is applied 0 = HIGH 1 = LOW
3	X	H_PULSE2_MODE: H Pulse 2 Mode. 0= Normal mode 1= Single-clock mode 0 = NORMAL 1 = ONE_CLOCK

21.8.20 DC_DISP_H_PULSE2_POSITION_A_0

H Pulse 2 Position A

Offset: 0x416 │ Byte Offset: 0x1058 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE2_END_A: H Pulse 2 End A (minimum -=PS_=-H_PULSE2_START_A+1)

Bit	Reset	Description
12:0	X	H_PULSE2_START_A: H Pulse 2 Start A (minimum 0)

21.8.21 DC_DISP_H_PULSE2_POSITION_B_0

H Pulse 2 Position B

Offset: 0x417 | Byte Offset: 0x105c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE2_END_B: H Pulse 2 End B (minimum -=PS_=-H_PULSE2_START_B+1)
12:0	X	H_PULSE2_START_B: H Pulse 2 Start B (minimum -=PS_=-H_PULSE2_END_A+1)

21.8.22 DC_DISP_H_PULSE2_POSITION_C_0

H Pulse 2 Position C

Offset: 0x418 | Byte Offset: 0x1060 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE2_END_C: H Pulse 2 End C (minimum -=PS_=-H_PULSE2_START_C+1)
12:0	X	H_PULSE2_START_C: H Pulse 2 Start C (minimum -=PS_=-H_PULSE2_END_B+1)

21.8.23 DC_DISP_H_PULSE2_POSITION_D_0

H Pulse 2 Position D

Offset: 0x419 | Byte Offset: 0x1064 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE2_END_D: H Pulse 2 End D (minimum -=PS_=-H_PULSE2_START_D+1)
12:0	X	H_PULSE2_START_D: H Pulse 2 Start D (minimum -=PS_=-H_PULSE2_END_C+1)

21.8.24 DC_DISP_V_PULSE0_CONTROL_0

V Pulse 0 Control

Vertical pulse 0 is a programmable pulse that repeats every frame.

This signal can have several pulses (A to C) per frame with programmable width as defined by the pairs of start and end positions. The pulses must not overlap and must occur in sequence: pulse A, then pulse B, etc. In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

The pulse generator processes the pairs of start and end position sequentially in the order of: Start A, End A, Start B, End B, etc. So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

This register specifies options for Vertical pulse 0.

Offset: 0x41a │ Byte Offset: 0x1068 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxx0000000000000xxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	0x0	V_PULSE0_H_POSITION: V Pulse 0 Horizontal Position. This parameter specifies the position where V Pulse 0 can toggle with respect to H reference point.
11:8	X	V_PULSE0_LAST: V Pulse 0 Last point 0= end on Start A position 1= end on End A position 2= end on Start B position 3= end on End B position 4= end on Start C position 5= end on End C position others= reserved 0 = START_A 1 = END_A 2 = START_B 3 = END_B 4 = START_C 5 = END_C
7:6	X	V_PULSE0_DELAY: V Pulse 0 Delay 0= no delay 1= 1-line delay 2= 2-line delay 3= reserved 0 = NODELAY 1 = DELAY1 2 = DELAY2
4	X	V_PULSE0_POLARITY: V Pulse 0 Polarity 0 = HIGH 1 = LOW

21.8.25 DC_DISP_V_PULSE0_POSITION_A_0

V Pulse 0 Position A

Offset: 0x41b │ Byte Offset: 0x106c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE0_END_A: V Pulse 0 End A (minimum -=PS_=-V_PULSE0_START_A+1)
12:0	X	V_PULSE0_START_A: V Pulse 0 Start A (minimum 0)

21.8.26 DC_DISP_V_PULSE0_POSITION_B_0

V Pulse 0 Position B

Offset: 0x41c │ Byte Offset: 0x1070 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE0_END_B: V Pulse 0 End B (minimum -=PS_=-V_PULSE0_START_B+1)
12:0	X	V_PULSE0_START_B: V Pulse 0 Start B (minimum -=PS_=-V_PULSE0_END_A+1)

21.8.27 DC_DISP_V_PULSE0_POSITION_C_0

V Pulse 0 Position C

Offset: 0x41d │ Byte Offset: 0x1074 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE0_END_C: V Pulse 0 End C (minimum -=PS_=-V_PULSE0_START_C+1)
12:0	X	V_PULSE0_START_C: V Pulse 0 Start C (minimum -=PS_=-V_PULSE0_END_B+1)

21.8.28 DC_DISP_V_PULSE1_CONTROL_0

V pulse 1 Control

Vertical pulse 1 is a programmable pulse that repeats every frame. This signal can have several pulses (A to C) per frame with programmable width as defined by the pairs of start and end positions. The pulses must not overlap and must occur in sequence: pulse A, then pulse B, etc.

In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

The pulse generator processes the pairs of start and end position sequentially in the order of: Start A, End A, Start B, End B, etc. So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

This register specifies options for Vertical pulse 1.

Offset: 0x41e │ Byte Offset: 0x1078 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxx0000000000000xxxxxxxxxxx0xxxx)

Bit	Reset	Description
28:16	0x0	V_PULSE1_H_POSITION: V Pulse 1 Horizontal Position. This parameter specifies the position where V Pulse 1 can toggle with respect to H reference point.
11:8	X	V_PULSE1_LAST: V pulse 1 Last point 0= end on Start A position 1= end on End A position 2= end on Start B position 3= end on End B position 4= end on Start C position 5= end on End C position others= reserved 0 = START_A 1 = END_A 2 = START_B 3 = END_B 4 = START_C 5 = END_C
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

Bit	Reset	Description
4	X	V_PULSE1_POLARITY: V pulse 1 Polarity 0 = HIGH 1 = LOW

21.8.29 DC_DISP_V_PULSE1_POSITION_A_0

V Pulse 1 Position A

Offset: 0x41f │ Byte Offset: 0x107c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE1_END_A: V Pulse 1 End A (minimum -=PS_=-V_PULSE1_START_A+1)
12:0	X	V_PULSE1_START_A: V Pulse 1 Start A (minimum 0)

21.8.30 DC_DISP_V_PULSE1_POSITION_B_0

V Pulse 1 Position B

Offset: 0x420 │ Byte Offset: 0x1080 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE1_END_B: V Pulse 1 End B (minimum -=PS_=-V_PULSE1_START_B+1)
12:0	X	V_PULSE1_START_B: V Pulse 1 Start B (minimum -=PS_=-V_PULSE1_END_A+1)

21.8.31 DC_DISP_V_PULSE1_POSITION_C_0

V Pulse 1 Position C

Offset: 0x421 │ Byte Offset: 0x1084 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE1_END_C: V Pulse 1 End C (minimum -=PS_=-V_PULSE1_START_C+1)
12:0	X	V_PULSE1_START_C: V Pulse 1 Start C (minimum -=PS_=-V_PULSE1_END_B+1)

21.8.32 DC_DISP_V_PULSE2_CONTROL_0

V Pulse 2 Control

Vertical pulse 2 is a programmable pulse that repeats every frame.

This signal can have one pulse (A) per frame with programmable width as defined by the pair of start and end positions.

In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

This register specifies options for Vertical pulse 2.

Offset: 0x422 | Byte Offset: 0x1088 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0000000000000xxxxxxxxxxx0xxxx)

Bit	Reset	Description
28:16	0x0	V_PULSE2_H_POSITION: V Pulse 2 Horizontal Position. This parameter specifies the position where V Pulse 2 can toggle with respect to the H reference point.
8	X	V_PULSE2_LAST: V pulse 2 Last point 0= end on Start A position 1= end on End A position others= reserved 0 = START_A 1 = END_A
4	0x0	V_PULSE2_POLARITY: V pulse 2 Polarity 0 = HIGH 1 = LOW

21.8.33 DC_DISP_V_PULSE2_POSITION_A_0

V Pulse 2 Position A

Offset: 0x423 | Byte Offset: 0x108c | Read/Write: R/W | Reset: 0x00000000 (0bxxx0000000000000xxxxxxxxxxx0xxxx)

Bit	Reset	Description
28:16	X	V_PULSE2_END_A: V Pulse 2 End A (minimum -=PS_=-V_PULSE2_START_A+1)
12:0	X	V_PULSE2_START_A: V Pulse 2 Start A (minimum 0)

21.8.34 DC_DISP_V_PULSE3_CONTROL_0

V Pulse 3 Control

Vertical pulse 3 is a programmable pulse that repeats every frame.

This signal can have one pulse (A) per frame with programmable width as defined by the pair of start and end positions.

In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

This register specifies options for Vertical pulse 3.

Offset: 0x424 | Byte Offset: 0x1090 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0000000000000xxxxxxxxxxx0xxxx)

Bit	Reset	Description
28:16	0x0	V_PULSE3_H_POSITION: V Pulse 3 Horizontal Position. This parameter specifies the position where V Pulse 3 can toggle with respect to H reference point.
8	X	V_PULSE3_LAST: V pulse 3 Last point 0= end on Start A position 1= end on End A position others= reserved 0 = START_A 1 = END_A

Bit	Reset	Description
4	0x0	V_PULSE3_POLARITY: V pulse 3 Polarity 0 = HIGH 1 = LOW

21.8.35 DC_DISP_V_PULSE3_POSITION_A_0

V Pulse 3 Position A

Offset: 0x425 │ Byte Offset: 0x1094 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE3_END_A: V Pulse 3 End A (minimum -=PS_=-V_PULSE3_START_A+1)
12:0	X	V_PULSE3_START_A: V Pulse 3 Start A (minimum 0)

21.8.36 DC_DISP_DISP_CLOCK_CONTROL_0

Display Clock Control

The shift clock divider is used to divide root clock for display controller module to generate internal shift clock for shifting data to the display. Output of this divider is typically used to generate the external shift clock which is sent to the display (SC0 and/or SC1) except for 1-pixel/1-clock parallel display.

The output of this divider is also used to generate Programmable Pulse (PP) signal. For 1-pixel/1-clock parallel display, SC0 and SC1 are generated using the output of pixel clock divider which can be set to 1, 2, or 4 for 1-pixel/1-clock parallel display.

The reason pixel clock divider 2 and 4 are allowed for 1-pixel/1-clock parallel display interface is so that the clock that generates PP can be generated with 2x or 4x higher frequency than pixel clock and therefore can produce higher resolution PP pulse positions. For all cases of parallel display, SC0 and SC1 can be further divided by 1, 2, or 4.

Class: Display Interface Settings

This register controls generation of shift clock to the display and internal pixel clock. Internal display pipeline runs with pixel clock and processes 1 pixel per clock.

Offset: 0x42e │ Byte Offset: 0x10b8 │ Read/Write: R/W │ Reset: 0x00000006 (0bxxxxxxxxxxxxxxxxxxxx000000000110)

Bit	Reset	SW Default	Description
11:8	0x0	PCD1	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

Bit	Reset	SW Default	Description
			5 = PCD6 6 = PCD8 7 = PCD9 8 = PCD12 9 = PCD16 10 = PCD18 11 = PCD24 12 = PCD13
7:0	0x6	NONE	SHIFT_CLK_DIVIDER: Shift Clock Divider. 0 = divide by 1 1 = divide by 1.5 2 = divide by 2 3 = divide by 2.5 4 = divide by 3 : : : : 254 = divide by 128 255 = divide by 128.5 Pixel clock divider is used to divide output of internal shift clock divider to generate internal pixel clock which is used to clock the internal horizontal and vertical counters. This divider also determines the output format for parallel interface, serial interface, and LCD SPI interface in conjunction with Display Data Format parameter. For 1-pixel/1-clock parallel display interface, valid settings are PCD1, PCD2, and PCD4. Note that the main reason to use PCD2 and PCD4 is to get higher frequency PP clock because the PP clock is always generated from the output of shift clock divider. For non 1-pixel/1-clock parallel display interface, valid settings are, PCD1H (2-pixel/3-clock), PCD2 (1-pixel/2-clock), and PCD3 (1-pixel/3-clock). For 1-channel serial display interface, valid settings are PCD3 (3-bpp 1-ch), PCD4 (3-bpp 1-ch), PCD6 (6-bpp 1-ch), PCD9 (9-bpp 1-ch), PCD12 (12-bpp 1-ch), PCD16 (16-bpp 1-ch), PCD18 (18-bpp 1-ch). For 2-channel serial display interface, valid settings are PCD2 (3-bpp 2-ch), PCD3 (6-bpp 2-ch), PCD6 (12-bpp 2-ch), PCD8 (16-bpp 2-ch), PCD9 (18-bpp 2-ch). For 3-channel serial display interface, valid settings are PCD1 (3-bpp 3-ch), PCD2 (6-bpp 3-ch), PCD3 (9-bpp 3-ch), PCD4 (12-bpp 3-ch), PCD6 (18-bpp 3-ch). For LCD SPI interface, valid settings are PCD12 (B4G4R4), PCD16 (B5G6R5), PCD18 (B6G6R6), PCD24 (B8G8R8), PCD8 (B5G6R5 with data/command bit), PCD6 (B5G6R5 with data/command start byte - depending on data/command bit), PCD4 (P8 for spi8), PCD9 (B5G6R5 with chip select deassertion at 8-bit boundary, spi16x2), PCD3 (P8 for spidc), PCD2 (B5G6R5 with data/command bit and chip select deassertion at 9-bit boundary, spi16x2dc), and PCD13 (spi12p2, no chip select deassertion between pairs of pixels).

21.8.37 DC_DISP_DISP_INTERFACE_CONTROL_0

Display Interface Control

This register specifies display interface options.

Note: Only the PCD1 setting is allowed.
Only default settings are allowed.

Offset: 0x42f │ Byte Offset: 0x10bc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00xxxx0000)

Bit	Reset	Description
9	0x0	DISP_DATA_ORDER: Display Data Order. This is effective only for 1-pixel/2-clock 16-/18-/24- bit parallel interface 0= Red pixel is output in the first clock and blue pixel is output in the second cycle 1= Blue pixel is output in the first clock cycle and red pixel is output in the second clock cycle 0 = RED_BLUE 1 = BLUE_RED
8	0x0	DISP_DATA_ALIGNMENT: Display Data Alignment. This is effective for parallel display data format and the associated Initialization Sequence (IS). 0= Output data is MSB-aligned. For 1-pixel/1-clock parallel display, the output data ordering is the same regardless of display Base Color Size. For 1-pixel/1-clock parallel display data alignment is optimized for 18-bpp so the 24-bit data ordering is: LD[5:0] are blue data bits 7-2 LD[11:6] are green data bits 7-2 LD[17:12] are red data bits 7-2 LD[19:18] are blue data bits 1-0 LD[21:20] are green data bits 1-0 LD[23:22] are red data bits 1-0 Note that LD18 to LD23 signals are multiplexed with control pins (see Pin Output Select definition) 1= Output data is LSB-aligned. For 1-pixel/1-clock parallel display the output data ordering is determined by display Base Color Size. For 1-pixel/1-clock parallel display data alignment is optimized for 24-bpp as follows: LD[7:0] are blue data bits 7-0 LD[15:8] are green data bits 7-0 LD[23:16] are red data bits 7-0 Note that LD18 to LD23 signals are multiplexed with control pins (see Pin Output Select definition) 0 = MSB 1 = LSB
3:0	0x0	DISP_DATA_FORMAT: Display Data Format Pixel Clock Divider is used together with this parameter to determine the exact display data format. 0 = DF1P1C : 0= 1-pixel/1-clock up to 24-bit parallel 1 = DF1P2C24B : 1= 1-pixel/2-clock 24-bit parallel 2 = DF1P2C18B : 2= 1-pixel/2-clock 18-bit parallel or 2-pixel/3-clock 12-bit parallel or 1-pixel/3-clock 18-bit parallel NOTE: for 2-pixel/3-clock 12-bit parallel, the horizontal display active time must be even number of pixels. 3 = DF1P2C16B : 3= 1-pixel/2-clock 16-bit parallel 4 = DF1S : 4= 1-channel serial NOTE: 1-/2-/3-channel serial display interface supported is a low-voltage differential serial interface. 5 = DF2S : 5= 2-channel serial 6 = DF3S : 6= 3-channel serial 7 = DFSPI : 7= SPI serial 8 = DF1P3C24B : 8= 1-pixel/3-clock 24-bit parallel 9 = DF2P1C18B : 9= 2-pixel/1-clock 18-bit parallel 10 = DFDUAL1P1C18B: 10= 1-pixel/1-clock 18-bit parallel (used for dual output)

21.8.38 DC_DISP_DISP_COLOR_CONTROL_0

Display Color Control

Offset: 0x430 | Byte Offset: 0x10c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx0000xxx0x0x0xx00xxxx00xx0000)

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
20	DISABLE	CMU_ENABLE: MD0-3 signals are general purpose mode signals that can be output in various pins (see Pin Output Select) to configure the display device. These bits are effective at start of frame. Typically these can be programmed in shadow register which takes effect on the next frame. 0 = DISABLE 1 = ENABLE
18	0x0	NON_BASE_COLOR: Non Base Color 0= zeros 1= ones
17	X	BLANK_COLOR: Blank Color 0= zeros 1= ones Non Base Color applies to least significant color bits which are not part of base color and it has higher priority over Border Color but lower priority over Blank color.
16	0x0	DISP_COLOR_SWAP: Display Color Swap 0= RGB (normal) 1= BGR (red-blue reverse) 0 = RGB 1 = BGR
13:12	0x0	ORD_DITHER_ROTATION: Ordered Dither Frame Rotation. This parameter specifies the rotation frequency of the dither matrix in terms of number of frames. If programmed to 0, there is no dither matrix rotation. If programmed to N, where N is larger than 0, the dither matrix is rotated clockwise every N frame.
9:8	X	DITHER_CONTROL: Dither Control 00= dither disabled 01= reserved 10= ordered dither 11= temporal dithering 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 = TEMPORAL

Bit	Reset	Description
7:6	0x0	TEMPORAL_DITHER_PHASE: Temporal dither LFSR phase control 2'b01: random_data <= 34'h3fffffff 2'b10: random_data <= 34'h155555555 2'b11: random_data <= 34'h2aaaaaaaa 2'b00: retains previous value Temporal dither supports only 8 to 6 bits dithering.
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

21.8.39 DC_DISP_COLOR_KEY0_LOWER_0

Color Key 0 Lower Value

Color Key 0 and Color Key 1

Two ranges of color key are defined and they are common for all windows because it is expected that typically only one window will have the color key enabled. Because there are two sets of color key, it is possible to have 2 windows each using one color key set.

Usage of this color key is described in the Display Color Key and Blending class.

Offset: 0x436 │ Byte Offset: 0x10d8 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	COLOR_KEY0_L_A: Color Key 0 Alpha (0xFF) Lower value
23:16	0x0	COLOR_KEY0_L_B: Color Key 0 Blue (U) Lower value
15:8	0x0	COLOR_KEY0_L_G: Color Key 0 Green (Y) Lower value
7:0	0x0	COLOR_KEY0_L_R: Color Key 0 Red (V) Lower value

21.8.40 DC_DISP_COLOR_KEY0_UPPER_0

Color Key 0 Upper Value

Offset: 0x437 | Byte Offset: 0x10dc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	COLOR_KEY0_U_A: Color Key 0 Alpha (0xFF) Upper value
23:16	0x0	COLOR_KEY0_U_B: Color Key 0 Blue (U) Upper value
15:8	0x0	COLOR_KEY0_U_G: Color Key 0 Green (Y) Upper value
7:0	0x0	COLOR_KEY0_U_R: Color Key 0 Red (V) Upper value

21.8.41 DC_DISP_COLOR_KEY1_LOWER_0

Color Key 1 Lower Value

Offset: 0x438 | Byte Offset: 0x10e0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	COLOR_KEY1_L_A: Color Key 1 Alpha (0xFF) Lower value
23:16	0x0	COLOR_KEY1_L_B: Color Key 1 Blue (U) Lower value
15:8	0x0	COLOR_KEY1_L_G: Color Key 1 Green (Y) Lower value
7:0	0x0	COLOR_KEY1_L_R: Color Key 1 Red (V) Lower value

21.8.42 DC_DISP_COLOR_KEY1_UPPER_0

Color Key 1 Upper Value

Offset: 0x439 | Byte Offset: 0x10e4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	COLOR_KEY1_U_A: Color Key 1 Alpha (0xFF) Upper value
23:16	0x0	COLOR_KEY1_U_B: Color Key 1 Blue (U) Upper value
15:8	0x0	COLOR_KEY1_U_G: Color Key 1 Green (Y) Upper value
7:0	0x0	COLOR_KEY1_U_R: Color Key 1 Red (V) Upper value

21.8.43 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.

In legacy cursor mode, only 32x32 and 64x64 are supported

Offset: 0x43c │ Byte Offset: 0x10f0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	CURSOR_FOREGROUND_B: Cursor Blue Foreground Color
15:8	X	CURSOR_FOREGROUND_G: Cursor Green Foreground Color
7:0	X	CURSOR_FOREGROUND_R: Cursor Red Foreground Color

21.8.44 DC_DISP_CURSOR_BACKGROUND_0

Cursor Background Color

Offset: 0x43d │ Byte Offset: 0x10f4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	CURSOR_BACKGROUND_B: Cursor Blue Background Color
15:8	X	CURSOR_BACKGROUND_G: Cursor Green Background Color
7:0	X	CURSOR_BACKGROUND_R: Cursor Red Background Color

21.8.45 DC_DISP_CURSOR_START_ADDR_0

Cursor Start Address

Offset: 0x43e │ Byte Offset: 0x10f8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:28	X	CURSOR_CLIPPING: Cursor Clipping Select 0 = DISPLAY : 00= display 1 = WA : 01= window A 2 = WB : 10= window B 3 = WC : 11= window C
25: 24	X	CURSOR_SIZE: Cursor Size 0 = C32X32 1 = C64X64 2 = C128X128 3 = C256X256
21:0	X	CURSOR_START_ADDR: Cursor Start Address bits 31:10

21.8.46 DC_DISP_CURSOR_START_ADDR_NS_0

Shadow of Cursor Start Address

Offset: 0x43f │ Byte Offset: 0x10fc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:28	X	CURSOR_CLIPPING_NS: Cursor Clipping Select 0 = DISPLAY : 00= display 1 = WA : 01= window A 2 = WB : 10= window B 3 = WC : 11= window C
25:24	X	CURSOR_SIZE_NS: Cursor Size 0 = C32X32 1 = C64X64 2 = C128X128 3 = C256X256
21:0	X	CURSOR_START_ADDR_NS: Cursor Start Address bits 31:10

21.8.47 DC_DISP_CURSOR_POSITION_0

Cursor Position

Cursor position is with respect to top-left corner of display active area, window A, window B, or window C as specified in the cursor clipping parameter.

Offset: 0x440 │ Byte Offset: 0x1100 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:16	X	V_CURSOR_POSITION: V cursor position (signed)
13:0	X	H_CURSOR_POSITION: H cursor position (signed)

21.8.48 DC_DISP_CURSOR_POSITION_NS_0

Shadow of Cursor Position

Offset: 0x441 │ Byte Offset: 0x1104 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:16	X	V_CURSOR_POSITION_NS: V cursor position (signed)
13:0	X	H_CURSOR_POSITION_NS: H cursor position (signed)

21.8.49 DC_DISP_DC_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control Register (where applicable) and Clock Gating Control

Note: The FIFO timing aspects of this register are no longer supported, but are retained for software compatibility. The clock override/ovr_mode fields of this register control the second level clock gating for the client and MC side of the MCCIF. All clock gating is enabled by default.

A ‘1’ written to the rclk/wclk override filed will result in one of the following:

- With wclk/rclk override mode = LEGACY, the clock reverts to the legacy mode of operation (where the clock is on whenever the client clock is enabled.
- With wclk/rclk override mode = ON, the clock is always on inside the MCCIF and PC.

A '1' written to the cclk override field keeps the client's clock always on inside the MCCIF.

Offset: 0x480 │ Byte Offset: 0x1200 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxx00000xxxxxxxxxxxx0000)

Bit	Reset	Description
20	LEGACY	DC_RCLK_OVR_MODE: 0 = LEGACY 1 = ON
19	LEGACY	DC_WCLK_OVR_MODE: 0 = LEGACY 1 = ON
18	0x0	DC_CCLK_OVERRIDE
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

21.8.50 DC_DISP_MCCIF_DISPLAY0A_HYST_0

Memory Client Hysteresis Control Register

Note: Hysteresis is no longer supported by the MCCIF clients. Registers are retained for software compatibility, but are not used by the hardware.

Offset: 0x481 │ Byte Offset: 0x1204 │ Read/Write: R/W │ Reset: 0xcf401f1f (0b11001111010000000001111100011111)

Bit	Reset	Description
31	ENABLE	CSR_DISPLAY0A2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CSR_DISPLAY0A2MC_HYST_REQ_TH
27:24	0xf	CSR_DISPLAY0A2MC_HYST_TM
23:16	0x38	CSR_DISPLAY0A2MC_DHYST_TH

Bit	Reset	Description
15:8	0x10	CSR_DISPLAY0A2MC_DHYST_TM
7:0	0x58	CSR_DISPLAY0A2MC_HYST_REQ_TM

21.8.51 DC_DISP_MCCIF_DISPLAY0B_HYST_0

Memory Client Hysteresis Control Register

Note: Hysteresis is no longer supported by the MCCIF clients. Registers are retained for software compatibility, but are not used by the hardware.

Offset: 0x482 | Byte Offset: 0x1208 | Read/Write: R/W | Reset: 0xcf401f1f (0b11001111010000000001111100011111)

Bit	Reset	Description
31	ENABLE	CSR_DISPLAY0B2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CSR_DISPLAY0B2MC_HYST_REQ_TH
27:24	0xf	CSR_DISPLAY0B2MC_HYST_TM
23:16	0x40	CSR_DISPLAY0B2MC_DHYST_TH
15:8	0x1f	CSR_DISPLAY0B2MC_DHYST_TM
7:0	0x1f	CSR_DISPLAY0B2MC_HYST_REQ_TM

21.8.52 DC_DISP_MCCIF_DISPLAY0C_HYST_0

Memory Client Hysteresis Control Register

Note: Hysteresis is no longer supported by the MCCIF clients. Registers are retained for software compatibility, but are not used by the hardware.

Offset: 0x483 | Byte Offset: 0x120c | Read/Write: R/W | Reset: 0xcf401f1f (0b11001111010000000001111100011111)

Bit	Reset	Description
31	ENABLE	CSR_DISPLAY0C2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CSR_DISPLAY0C2MC_HYST_REQ_TH
27:24	0xf	CSR_DISPLAY0C2MC_HYST_TM
23:16	0x40	CSR_DISPLAY0C2MC_DHYST_TH
15:8	0x1f	CSR_DISPLAY0C2MC_DHYST_TM
7:0	0x1f	CSR_DISPLAY0C2MC_HYST_REQ_TM

21.8.53 DC_DISP_DISP_MISC_CONTROL_0

Miscellaneous Controls

Offset: 0x4c1 | Byte Offset: 0x1304 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx10)

Bit	Reset	Description
1	0x1	UF_LINE_FLUSH: Enable underflow line flush as opposed to end-of-frame flush. underflow line flush 0 = DISABLE; 1 = ENABLE
0	0x0	PHASE_SHIFT_2P1C18B: Enable phase shift for 2P1C format phase shift SC0/SC1 will be delayed for one pixel clock cycle. In 2P1C format, data will hold for 2 pixel clocks, so either choice should work 0 = DISABLE 1 = ENABLE

21.8.54 DC_DISP_SD_CONTROL_0

The Smart Dimmer (PRISM 3) function takes advantage of the fact that a perceived pixel brightness in an LCD depends on both the pixel brightness value and the backlight intensity to reduce the backlight intensity to save power. Statistics are gathered on the current video frame and an "enhancement" is applied to subsequent frames that increases the pixel brightness value and reduces the backlight brightness to give an overall image intensity that is mostly the same as before.

PRISM 3 Control

Offset: 0x4c2 | Byte Offset: 0x1308 | Read/Write: R/W | Reset: 0x00004000 (0bx000xxxxxxxxxxx0100000000000000)

Bit	Reset	Description
30:29	BIAS0	K_INIT_BIAS: BIAS of the initial K bias MSB of count 0 = BIAS0 : bias 0 1 = BIAS1 : bias 1.0 2 = BIAS_HALF : bias 0.5 3 = BIAS_MSB
28	VSYNC	SD_FRAME_PROC_CONTROL: when to run per-frame processing. VSYNC is legacy behavior. 0 = VSYNC 1 = VPULSE2
15	DISABLE	SMOOTH_K_ENABLE: When enable, maximum raw K change per frame is limited to SMOOTH_K_INCR 0 = DISABLE 1 = ENABLE
14	ENABLE	SOFT_CLIPPING_ENABLE: When enabled, enhancement gain (K) is reduced for pixels above SOFT_CLIPPING_THRESHOLD level to avoid saturation (clipping). 0 = DISABLE 1 = ENABLE
13	DISABLE	SD_WINDOW_ENABLE: When enabled, constrain histogram (and therefore backlight) to a rectangular subset of display. The rectangle has upper/left corner described by SD_WINDOW_POSITION, and width/height by SD_WINDOW_SIZE. Useful for when display has regions that should not influence SD, such as letterbox borders, etc. Enhancement is always performed on whole display, regardless. DISABLE: SD histogram done over whole display ENABLE: SD histogram done over rectangular subset given by SD_WINDOW* registers 0 = DISABLE 1 = ENABLE
12	DISABLE	K_LIMIT_ENABLE: When enabled, Max. K is taken from K_LIMIT register rather than computed from AGGRESSIVENESS. DISABLE: K is limited by AGGRESSIVENESS ENABLE: K is limited by K_LIMIT register. 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
11	0x0	SD_CORRECTION_MODE: Determines which K values are used to modify the pixel values. MANUAL : The K values in the SD_MAN_K_VALUES register are used to modify pixel values. 0 = AUTO_CORRECT : AUTO_CORRECT : SD Hardware computed K values are used to 1 = MANUAL
10	0x0	SD_ONE_SHOT: Enables the PRISM 3 function for one frame only. on which the SD statistics are gathered. NOTE: The SD_ENABLE field (see above) must be set to ONE_SHOT in order to use this function. 0 = DISABLE : Automatically cleared to DISABLE at the end of the frame 1 = ENABLE
9:8	0x0	HW_UPDATE_DLY: Determines the delay - in video frames - of the update of the hardware enhancement value that is applied to the pixels. This is useful for allowing the software some time to update the backlight control, when the control must be sent via side-band control packets or by some other means of control that incurs a sizeable delay. Being able to delay the hardware update ensures that the modification of the pixels occurs as nearly simultaneously with the update of the backlight as possible. Value Description 0 No delay - pixels modified immediately 1 New enhancement value delayed by 1 frame. 2 New enhancement value delayed by 2 frames. 3 New enhancement value delayed by 3 frames.
7:5	0x0	AGGRESSIVENESS: The "aggressiveness" level of the PRISM 3 algorithm. Higher aggressiveness levels result in higher power savings at the potential expense of image quality. The number programmed determines how many highlight pixels will be allowed to exceed the maximum representable brightness value and be clipped to that value. It also determines the maximum allowed enhancement value (k) applied to the pixel brightness. AGGRESSIVENESS Description % pixels Max. k value crushed value 0 Essentially off 0% 1.00 1 Highest quality < 5% 1.10 2 Higher quality < 10% 1.15 3 Balanced < 15% 1.20 4 Higher battery life < 20% 1.25 5 Highest battery life < 25% 1.50
4:3	0x0	BIN_WIDTH: Width of the Histogram bins, in quantization levels. EIGHT = 8 levels per bin. Bins span range from 0 to 255 0 = ONE : ONE = 1 level per bin. Bins span range from 224 to 255 1 = TWO : TWO = 2 levels per bin. Bins span range from 192 to 255 2 = FOUR : FOUR = 4 levels per bin. Bins span range from 128 to 255 3 = EIGHT
2	0x0	USE_VID_LUMA: Use Video Luminance control of luminance: Luminance = MAX(R, G, B) ENABLE = use "video" luminance, which is determined by the coefficients in the SD_CSC_COEFFS register See the SD_CSC_COEFFS register for details. 0 = DISABLE : DISABLE = use Hue Saturation Value (HSV) version 1 = ENABLE
1:0	0x0	SD_ENABLE: Enables the PRISM 3 Function ONE_SHOT = SD function enabled, but statistics gathering limited to the next frame only. 0 = DISABLE : DISABLE = SD function disabled. 1 = ENABLE : ENABLE = SD function enabled and operational. 2 = ONE_SHOT

21.8.55 DC_DISP_SD_CSC_COEFF_0

Luminance calculation coefficients used to convert the red, green, and blue color components into a luminance value. The conversion is performed according to the following equation: Luminance = (R * R_COEFF + G * G_COEFF + B * B_COEFF) >> 4 It is suggested that the values of the coefficients be programmed as shown below, though user-defined color spaces are also accommodated. Color Space R_COEFF G_COEFF B_COEFF ITU-R Bt601 5 9 2 ITU-R Bt709 3 12 1. The coefficients

do not have to be particularly accurate, hence their low precision, and the coefficients used are open to experimentation to obtain the best results.

Offset: 0x4c3 | Byte Offset: 0x130c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx0000xxxx0000xxxx0000xxxx)

Bit	Reset	Description
23:20	0x0	B_COEFF: Blue coefficient for luminance calculation
15:12	0x0	G_COEFF: Green coefficient for luminance calculation
7:4	0x0	R_COEFF: Red coefficient for luminance calculation

21.8.56 DC_DISP_SD_LUT_0

Enhancement value (k) Look Up Table. Each LUT entry contains the value of k for each of the three color components. Since the value of k for the color components must be the reciprocal of the hardware-computed value of k, and the hardware value is guaranteed to be less than or equal to 1, the values in the LUT represent the fractional part of k, with an implied 1 to the left of the decimal place. For example, if the programmed value of R_LUT was 64 (01000000 in binary), then the actual value of k generated would be 1.01000000 in binary or 1.25 in decimal. To program a default, linear response into the LUT, use the following code as a guide: for (i = 0; i < 9; i++) { t = (4096 / (8 + i)) - 256; if (t > 255) t = 255; R_LUT[i] = t; G_LUT[i] = t; B_LUT[i] = t; } For other non-linear response curves (for example, to take display gamma into consideration), this code will have to be modified.
This is an array of 9 identical register entries; the register fields below apply to each entry.

Offset: 0x4c4..0x4cc | Byte Offset: 0x1310..0x1330 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	B_LUT: Blue Enhancement value (k) Look Up Table
15:8	0x0	G_LUT: Green Enhancement value (k) Look Up Table
7:0	0x0	R_LUT: Red Enhancement value (k) Look Up Table

21.8.57 DC_DISP_SD_FLICKER_CONTROL_0

Flicker Reduction Control Register

The flicker control prevents rapid and frequent changes in the enhancement value.

Offset: 0x4cd | Byte Offset: 0x1334 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	THRESHOLD: The amount by which the currently calculated enhancement value must deviate from the currently active enhancement value for it to increment the TIME_LIMIT counter.
7:0	0x0	TIME_LIMIT: Length of time - in frames - that the enhancement value must deviate from the current value by more than THRESHOLD, before the enhancement value changes.

21.8.58 DC_DISP_SD_PIXEL_COUNT_0

Status / debug register showing the total number of active pixels

Offset: 0x4ce | Byte Offset: 0x1338 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	NUM_PIXELS: in the preceding output frame. Expressed as a quantity of 256 pixels. In other words, a 640 x 480 image has 307200 pixels. The value in this register would be 307200 /

Bit	Reset	Description
		256 = 1200

21.8.59 DC_DISP_SD_HISTOGRAM_0

Status/debug registers showing the gathered histogram data. Each register contains 4 histogram bins, for a total of 8 x 4 = 32 bins. Each bin has been approximately scaled to the number of pixels in the image so that a single quantization step in a bin represents a fraction of between 1/256 and 1/128 of the total number of pixels in the image.

This is an array of 8 identical register entries; the register fields below apply to each entry.

Offset: 0x4cf..0x4d6 │ Byte Offset: 0x133c..0x1358 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	BIN_3
23:16	X	BIN_2
15:8	X	BIN_1
7:0	X	BIN_0

21.8.60 DC_DISP_SD_BL_PARAMETERS_0

Backlight response parameters. Defines the parameters for the backlight temporal response model.

Offset: 0x4d7 │ Byte Offset: 0x135c │ Read/Write: R/W │ Reset: 0x00000400 (0bxxxxxxxx00000000xxxxx10000000000) | Default: 0x00ff0000

Bit	Reset	SW Default	Description
23:16	0x0	0xff	STEP: Determines the instantaneous portion of the target value of enhancement that is applied. 0 = 0% : response is entirely exponential and determined by TIME_CONSTANT 128 = 50% : response will instantly step up by 50% and will then be exponential. 255 = 100% : response is entirely instantaneous. TIME_CONSTANT has no effect.
10:0	0x400	NONE	TIME_CONSTANT: The time constant for the response curve. This value represents the fraction by which the value of enhancement value approaches the target value each frame. Example values are shown below: 0 : The value will never reach the target (infinite TC) 512 : The next value will be halfway between the current value and the target value. 1024 : The next value will be 100% of the target value. In other words - an instantaneous response.

21.8.61 DC_DISP_SD_BL_TF_0

Backlight Transfer Function

Each register contains 4 points on the Transfer Function curve that defines how the backlight output changes with respect to the control input. Each point defines a value at the vertex of a 16 segment line. The 17th point is defined to be the maximum value (it is assumed 100% control == 100% light output).

This is an array of 4 identical register entries; the register fields below apply to each entry.

Offset: 0x4d8..0x4db │ Byte Offset: 0x1360..0x136c │ Read/Write: R/W │Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	POINT_3
23:16	X	POINT_2
15:8	X	POINT_1
7:0	X	POINT_0

21.8.62 DC_DISP_SD_BL_CONTROL_0

Backlight Control Register

Offset: 0x4dc │ Byte Offset: 0x1370 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	R/W	Reset	Description
15:8	RO	X	BRIGHTNESS: Backlight brightness modification value. This value is determined by the hardware according to all the other control registers and the image content. The amount by which the backlight should be modified is given as a fraction with 0 representing that the backlight should be off and 255 representing no change in the backlight intensity. Other values vary linearly between these two extremes. New BL control = (Old BL control * BRIGHTNESS) / 255
1:0	RW	0x0	BL_MODE: Control Mode: and adjust the backlight brightness itself. PWM_AUTO : Hardware will adjust the backlight PWM control signal directly using the value in BRIGHTNESS. * OTHER VALUES ARE RESERVED FOR FUTURE USE * 0 = MANUAL : MANUAL : Hardware makes no BL corrections. Software must read the BRIGHTNESS field 1 = PWM_AUTO

21.8.63 DC_DISP_SD_HW_K_VALUES_0

Hardware-computed values of K for each color component. These values are used to modify the pixel values when CORRECTION_MODE in the SD_CONTROL register is set to AUTO_CORRECT. The numerical format is a fractional 10-bit number. Since we only ever want to make the pixel values bigger, there is an implied integer "1" not present in the register value. Only the fractional part is returned. Also in Tegra K1 devices, only the top 7 bits of the 10-bit K word have any meaning. The bottom 3 bits are always 0 and are reserved for future expansion. This register is read only.

Offset: 0x4dd │ Byte Offset: 0x1374 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:20	X	HW_K_BLUE: Value of K for blue pixels.
19:10	X	HW_K_GREEN: Value of K for green pixels.
9:0	X	HW_K_RED: Value of K for red pixels.

21.8.64 DC_DISP_SD_MAN_K_VALUES_0

Manual values of K for each color component. These values are used to modify the pixel values when CORRECTION_MODE in the SD_CONTROL register is set to MANUAL. The numerical format is a fractional 10-bit number. Since we only ever want to make the pixel values bigger, there is an implied integer "1" not present in the register value. Only the fractional part can be programmed. Therefore, K values can only be programmed from 1.0 to slightly less than 2.0 .Also note that only the top 7 bits of the 10-bit K word have any effect in the hardware. The bottom 3 bits are reserved for future expansion.

Offset: 0x4de | Byte Offset: 0x1378 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:20	X	MAN_K_BLUE: Value of K for blue pixels.
19:10	X	MAN_K_GREEN: Value of K for green pixels.
9:0	X	MAN_K_RED: Value of K for red pixels.

21.8.65 DC_DISP_SD_K_LIMIT_0

Offset: 0x4df | Byte Offset: 0x137c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	K_LIMIT: When K_LIMIT_ENABLE=ENABLE, limits raw K independently of AGGRESSIVENESS. Currently, only the bottom 8 bits are effective. The upper 2 are reserved for future expansion.

21.8.66 DC_DISP_SD_WINDOW_POSITION_0

SD Window Position

Offset: 0x4e0 | Byte Offset: 0x1380 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	SD_WIN_V_POSITION: SD window vertical position. This is specified with respect to the top edge of active display area.
12:0	X	SD_WIN_H_POSITION: SD window horizontal position (pixels). This is specified with respect to the left edge of active display area.

21.8.67 DC_DISP_SD_WINDOW_SIZE_0

SD Window Size

Offset: 0x4e1 | Byte Offset: 0x1384 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	SD_WIN_V_SIZE: SD window vertical height (lines).
12:0	X	SD_WIN_H_SIZE: SD window horizontal width (pixels).

21.8.68 DC_DISP_SD_SOFT_CLIPPING_0

SD soft clipping parameters. Software programs both threshold and a reciprocal thereof.

Offset: 0x4e2 | Byte Offset: 0x1388 | Read/Write: R/W | Reset: 0x02000080 (0b00000010000000000xxxxxxx10000000)

Bit	Reset	Description
31:16	0x200	SOFT_CLIPPING_RECIP: Reciprocal of inverse threshold. Compute as RECIP = int(64*1024 * 1.0 / (256 - THRESHOLD)). For example, for THRESHOLD=128, RECIP = int(64*1024 * 1.0 / (256 - 128)) = 512.
7:0	0x80	SOFT_CLIPPING_THRESHOLD: Threshold at which pixel enhancement gain is reduced.

21.8.69 DC_DISP_SD_SMOOTH_K_0

Offset: 0x4e3 | Byte Offset: 0x138c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
13:0	X	SMOOTH_K_INCR: When SMOOTH_K_ENABLE=1, the raw K is changed at most by SMOOTH_K_INCR per frame. 8.6 fixed-point fraction, with resolution 1/64. Currently, only the top 12 bits are effective. The bottom 2 are reserved for future expansion, so the effective resolution is 1/16.

21.8.70 DC_DISP_BLEND_BACKGROUND_COLOR_0

Offset: 0x4e4 | Byte Offset: 0x1390 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	BKGND_ALPHA
23:16	0x0	BKGND_BLUE
15:8	0x0	BKGND_GREEN
7:0	0x0	BKGND_RED

21.8.71 DC_DISP_INTERLACE_CONTROL_0

1080i Related Register

This register controls whether interlacing is enabled or disabled. When interlacing is enabled, the data is fetched from memory at 1080i (interlaced) and all fields are passed directly to the panel. A set of even (normal) and odd (new for interlacing) registers define the field base addresses and timings.

If interlacing is disabled, the following interlacing related registers will have NO effect.

Offset: 0x4e5 | Byte Offset: 0x1394 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	R/W	Reset	Description
2	RO	X	INTERLACE_STATUS: 0 = field1 1 = field2 0 = FIELD1 1 = FIELD2
1	RW	0x0	INTERLACE_START: 0 = field1 1 = field2 0 = FIELD1 1 = FIELD2
0	RW	0x0	INTERLACE_ENABLE: 0 = DISABLE 1 = ENABLE

21.8.72 DC_DISP_INTERLACE_FIELD2_REF_TO_SYNC_0

1080i Related Register

This register controls the ref to sync offset for H sync and V sync for FIELD2.

Offset: 0x4e6 | Byte Offset: 0x1398 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	FIELD2_V_REF_TO_SYNC: V reference to VSYNC for FIELD2. IMP - Units are in pixel clocks. Generally, for the non-interlaced timing, the units are in line clocks; however, for interlaced V sync must be generated at a pixel granularity with respect to V ref.
12:0	X	FIELD2_H_REF_TO_SYNC: H reference to HSYNC for FIELD2. Units are in pixel clocks.

21.8.73 DC_DISP_INTERLACE_FIELD2_SYNC_WIDTH_0

1080i Related Register

This register controls the H and V sync widths for FIELD2.

Offset: 0x4e7 | Byte Offset: 0x139c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	FIELD2_V_SYNC_WIDTH: This field controls the V sync widths for FIELD2. Units are in line clocks.
12:0	X	FIELD2_H_SYNC_WIDTH: This field controls the H sync widths for FIELD2. Units are in pixel clocks.

21.8.74 DC_DISP_INTERLACE_FIELD2_BACK_PORCH_0

1080i Related Register

This register controls the H and V back porch widths for FIELD2.

Offset: 0x4e8 | Byte Offset: 0x13a0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	FIELD2_V_BACK_PORCH: V back porch. Units are in line clocks.
12:0	X	FIELD2_H_BACK_PORCH: H back porch. Units are in pixel clocks.

21.8.75 DC_DISP_INTERLACE_FIELD2_FRONT_PORCH_0

1080i Related Register

This register controls the H and V front porch widths for FIELD2.

Offset: 0x4e9 | Byte Offset: 0x13a4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	FIELD2_V_FRONT_PORCH: V front porch. Units are in line clocks.
12:0	X	FIELD2_H_FRONT_PORCH: H front porch. Units are in pixel clocks.

21.8.76 DC_DISP_INTERLACE_FIELD2_DISP_ACTIVE_0

1080i Related Register

This register controls the H and V active widths for FIELD2.

Offset: 0x4ea | Byte Offset: 0x13a8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	FIELD2_V_DISP_ACTIVE: V active width. Units are in line clocks.
12:0	X	FIELD2_H_DISP_ACTIVE: H active width. Units are in pixel clocks.

21.8.77 DC_DISP_CURSOR_UNDERFLOW_CTRL_0

Offset: 0x4eb | Byte Offset: 0x13ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxx0)

Bit	Reset	Description
7	0x0	CURSOR_UFLOW_CYA: 0 = DISABLE 1 = ENABLE
0	0x0	CURSOR_UFLOW_CTRL_DBG_MODE: 0 = DISABLE 1 = ENABLE

21.8.78 DC_DISP_CURSOR_START_ADDR_HI_0

Cursor Start Address

Offset: 0x4ec | Byte Offset: 0x13b0 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	CURSOR_START_ADDR_HI: Cursor Start Address bits 33:32

21.8.79 DC_DISP_CURSOR_START_ADDR_HI_NS_0

Shadow of Cursor Start Address

Offset: 0x4ed | Byte Offset: 0x13b4 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	CURSOR_START_ADDR_HI_NS

21.8.80 DC_DISP_CURSOR_INTERLACE_CONTROL_0

This register controls whether interlacing is enabled or disabled. If interlacing is disabled, the interlacing related registers will have NO effect.

Cursor Start Address bits 33:32

Offset: 0x4ee | Byte Offset: 0x13b8 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00x00)

Bit	R/W	Reset	Description
4	RW	0x0	CURSOR_INTERLACE_FIELD2_VOFF_INCR: FIELD2 v position incr Enable 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
3	RW	0x0	CURSOR_INTERLACE_FIELD1_VOFF_INCR: FIELD1 v position incr Enable 0 = DISABLE 1 = ENABLE
2	RO	X	CURSOR_INTERLACE_STATUS: Current line start. 0= line 0 1= line 1 0 = LINE0 1 = LINE1
1	RW	0x0	CURSOR_INTERLACE_START: Starting line 0= line 0 1= line 1 0 = LINE0 1 = LINE1
0	RW	0x0	CURSOR_INTERLACE_ENABLE: Interlace enable 0 = DISABLE 1 = ENABLE

21.8.81 DC_DISP_CSC2_CONTROL_0

The CSC2 can be used for RGB to YCbCr conversion used in 1080i HDMI output. This hardcoded color space converter supports YCBCR709 and YCBCR601 output as YUV444.

Offset: 0x4ef | Byte Offset: 0x13bc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	LIMIT_RGB_COLOR: scale RGB [0,255] to [16,235] 0 = DISABLE 1 = ENABLE
1:0	RGB	OUTPUT_COLOR_SELECT: output color select: RGB for normal output, YCBCR709 for HDMI or YCBCR601 for older television color formats. 0 = RGB 1 = YCBCR709 2 = YCBCR601

21.8.82 DC_DISP_BLEND_CURSOR_CONTROL_0

Offset: 0x4f1 | Byte Offset: 0x13c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx0xxxxxx00xxxxxx0000000000)

Bit	Reset	Description
24	0x0	CURSOR_MODE_SELECT: 0 = LEGACY 1 = NORMAL
17:16	0x0	CURSOR_DST_BLEND_FACTOR_SELECT: 0 = ZERO 1 = K1 2 = NEG_K1_TIMES_SRC
9:8	0x0	CURSOR_SRC_BLEND_FACTOR_SELECT: 0 = K1 1 = K1_TIMES_SRC
7:0	0x0	CURSOR_ALPHA

21.8.83 DC_DISP_DVFS_CURSOR_CONTROL_0

Offset: 0x4f2 | Byte Offset: 0x13c8 | Read/Write: R/W | Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000011)

Bit	Reset	Description
8	DISABLE	CURSOR_DVFS_ENABLE: DISABLE: Forces ready_for_latency_event true (i.e., always allows DVFS event) ENABLE: Allows ready_for_latency_event to toggle, based on CURSOR_DVFS_THRESHOLD 0 = DISABLE 1 = ENABLE
7:0	0x3	CURSOR_DVFS_THRESHOLD

21.8.84 DC_DISP_CURSOR_UFLOW_DBG_PIXEL_0

Offset: 0x4f3 | Byte Offset: 0x13cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CURSOR_UFLOW_DBG_PIXEL

21.8.85 DC_DISP_CURSOR_SPOOLUP_CONTROL_0

Programmable spool up for cursor. If spoolup count > hc_vactive_start, it will be clamped to zero.

Offset: 0x4f4 | Byte Offset: 0x13d0 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000001)

Bit	Reset	Description
7:0	0x1	CURSOR_SPOOLUP_START

21.8.86 DC_DISP_DISPLAY_CLK_GATE_OVERRIDE_0

Offset: 0x4f5 | Byte Offset: 0x13d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	CMU_CLK_GATE_OVERRIDE: Disable clock-gating of the CMU module. 0 = DISABLE 1 = ENABLE
0	0x0	CURSOR_CLK_GATE_OVERRIDE: Disable clock-gating of cursor memfetch/control modules 0 = DISABLE 1 = ENABLE

21.8.87 DC_DISP_DISPLAY_DBG_TIMING_0

Offset: 0x4f6 | Byte Offset: 0x13d8 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	H_BLANK
28:16	X	H_COUNT
15	X	V_BLANK
12:0	X	V_COUNT

21.8.88 DC_DISP_DISPLAY_SPARE0_0

Offset: 0x4f7 | Byte Offset: 0x13dc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH_0

21.8.89 DC_DISP_DISPLAY_SPARE1_0

Offset: 0x4f8 | Byte Offset: 0x13e0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH_1

21.9 Window A (WINC_A) Registers

These registers control window A parameters.

> **Note:** There are three copies of these registers for windows A, B, and C. Windows A, B, and C support different features. Windows D, H, and T have subsets of these registers.

Windows A/B/C: color palette, digital vibrance, color space conversion, horizontal/vertical filtering

Window T: only 2BPP and 4BPP pitched linear format. None of the features listed in Windows A/B/C. The registers under DC_WINT are only accessed directly and not triple buffered.

21.9.1 DC_WINC_A_COLOR_PALETTE_0

This is used for palletized data format (color depth of 8 bpp or less) or for gamma correction for non-palletized data formats (color depth of more than 8-bpp).

Each window has its own color palette which consists of three 256x8 register files which can be written by the host and indexed (read) by the window.

For palletized data formats less than 8 bpp, the pixel data is aligned to least significant bits of the palette index (address), and the remaining upper bits are filled with the corresponding bits of the Palette Color Extension. For example, for 4-bpp mode, the pixel data occupies bits 3:0 of the palette index, and bits 7:4 of the palette index are set to bits 7:4 of the Palette Color Extension.

Note that a host read is assumed to be not needed - software can cache the color palette in system memory.

This is an array of 256 identical register entries; the register fields below apply to each entry.

Window A Color Palette

Offset: 0x500..0x5ff | Byte Offset: 0x1400..0x17fc | Read/Write: WO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	A_COLOR_PALETTE_B: Blue Color Palette
15:8	X	A_COLOR_PALETTE_G: Green Color Palette
7:0	X	A_COLOR_PALETTE_R: Red Color Palette

21.9.2 DC_WINC_A_PALETTE_COLOR_EXT_0

Palette extension for 1-bpp, 2-bpp, and 4-bpp. These bits provide the upper most significant bits for indexing the color palette.

Window A Palette Color Extension

Offset: 0x600 │ Byte Offset: 0x1800 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:1	X	A_PALETTE_COLOR_EXT: Window A Palette Color Extension. Bits 7:1 are used for 1-bpp mode, bits 7:2 are used for 2-bpp mode, and bits 7:4 are used for 4-bpp mode

21.9.3 DC_WINC_A_H_FILTER_P00_0

Horizontal Scaling Filter Coefficients

Horizontal scaling filter is a 6-tap filter with 4-bit positional phase.

- Coefficients 0 and 5 are 3-bit signed values ranging from -4 to 3.
- Coefficients 1 and 4 are 5-bit signed values ranging from -16 to 15.
- Coefficients 2 and 3 are 8-bit unsigned values ranging from 0 to 128.
- Coefficient 0 is the multiplier for the earliest pixel (P0) in the group of 6 pixels and
- Coefficient 5 is the multiplier for the latest pixel (P5) in the group. The output pixel positional phase is defined as centered in P2 if the positional phase is 0 or proportionally in between P2 and P3 if the positional phase is larger than 0.

The sum of all coefficients for each phase should be 128 typically, and software should never program the sum of all coefficients for a phase to be more than 128.

For each horizontal positional phase, the 6 filter coefficients require 32 register bits.

Note that color value ranges from 0 to 255 for Y, R, G, B, and -128 to 127 for U and V.

Window A Horizontal Filter phase 00

Offset: 0x601 │ Byte Offset: 0x1804 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P00C5: Phase 00 coefficient 5 (typically 0)
28:24	X	A_H_FILTER_P00C4: Phase 00 coefficient 4 (typically 0)
23:16	X	A_H_FILTER_P00C3: Phase 00 coefficient 3 (typically 0)
15:8	X	A_H_FILTER_P00C2: Phase 00 coefficient 2 (typically 128)
7:3	X	A_H_FILTER_P00C1: Phase 00 coefficient 1 (typically 0)
2:0	X	A_H_FILTER_P00C0: Phase 00 coefficient 0 (typically 0)

21.9.4 DC_WINC_A_H_FILTER_P01_0

Window A Horizontal Filter phase 01

Offset: 0x602 | Byte Offset: 0x1808 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P01C5: Phase 01 coefficient 5 (typically 1)
28:24	X	A_H_FILTER_P01C4: Phase 01 coefficient 4 (typically -2)
23:16	X	A_H_FILTER_P01C3: Phase 01 coefficient 3 (typically 8)
15:8	X	A_H_FILTER_P01C2: Phase 01 coefficient 2 (typically 124)
7:3	X	A_H_FILTER_P01C1: Phase 01 coefficient 1 (typically -4)
2:0	X	A_H_FILTER_P01C0: Phase 01 coefficient 0 (typically 1)

21.9.5 DC_WINC_A_H_FILTER_P02_0

Window A Horizontal Filter phase 02

Offset: 0x603 | Byte Offset: 0x180c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P02C5: Phase 02 coefficient 5 (typically 1)
28:24	X	A_H_FILTER_P02C4: Phase 02 coefficient 4 (typically -5)
23:16	X	A_H_FILTER_P02C3: Phase 02 coefficient 3 (typically 17)
15:8	X	A_H_FILTER_P02C2: Phase 02 coefficient 2 (typically 122)
7:3	X	A_H_FILTER_P02C1: Phase 02 coefficient 1 (typically -8)
2:0	X	A_H_FILTER_P02C0: Phase 02 coefficient 0 (typically 1)

21.9.6 DC_WINC_A_H_FILTER_P03_0

Window A Horizontal Filter phase 03

Offset: 0x604 | Byte Offset: 0x1810 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P03C5: Phase 03 coefficient 5 (typically 2)
28:24	X	A_H_FILTER_P03C4: Phase 03 coefficient 4 (typically -7)
23:16	X	A_H_FILTER_P03C3: Phase 03 coefficient 3 (typically 27)
15:8	X	A_H_FILTER_P03C2: Phase 03 coefficient 2 (typically 115)
7:3	X	A_H_FILTER_P03C1: Phase 03 coefficient 1 (typically -11)
2:0	X	A_H_FILTER_P03C0: Phase 03 coefficient 0 (typically 2)

21.9.7 DC_WINC_A_H_FILTER_P04_0

Window A Horizontal Filter phase 04

Offset: 0x605 | Byte Offset: 0x1814 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P04C5: Phase 04 coefficient 5 (typically 2)
28:24	X	A_H_FILTER_P04C4: Phase 04 coefficient 4 (typically -9)
23:16	X	A_H_FILTER_P04C3: Phase 04 coefficient 3 (typically 37)
15:8	X	A_H_FILTER_P04C2: Phase 04 coefficient 2 (typically 109)
7:3	X	A_H_FILTER_P04C1: Phase 04 coefficient 1 (typically -13)
2:0	X	A_H_FILTER_P04C0: Phase 04 coefficient 0 (typically 2)

21.9.8 DC_WINC_A_H_FILTER_P05_0

Window A Horizontal Filter phase 05

Offset: 0x606 | Byte Offset: 0x1818 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P05C5: Phase 05 coefficient 5 (typically 2)
28:24	X	A_H_FILTER_P05C4: Phase 05 coefficient 4 (typically -11)
23:16	X	A_H_FILTER_P05C3: Phase 05 coefficient 3 (typically 47)
15:8	X	A_H_FILTER_P05C2: Phase 05 coefficient 2 (typically 102)
7:3	X	A_H_FILTER_P05C1: Phase 05 coefficient 1 (typically -15)
2:0	X	A_H_FILTER_P05C0: Phase 05 coefficient 0 (typically 3)

21.9.9 DC_WINC_A_H_FILTER_P06_0

Window A Horizontal Filter phase 06

Offset: 0x607 | Byte Offset: 0x181c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P06C5: Phase 06 coefficient 5 (typically 3)
28:24	X	A_H_FILTER_P06C4: Phase 06 coefficient 4 (typically -13)
23:16	X	A_H_FILTER_P06C3: Phase 06 coefficient 3 (typically 56)
15:8	X	A_H_FILTER_P06C2: Phase 06 coefficient 2 (typically 94)
7:3	X	A_H_FILTER_P06C1: Phase 06 coefficient 1 (typically -15)
2:0	X	A_H_FILTER_P06C0: Phase 06 coefficient 0 (typically 3)

21.9.10 DC_WINC_A_H_FILTER_P07_0

Window A Horizontal Filter phase 07

Offset: 0x608 | Byte Offset: 0x1820 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P07C5: Phase 07 coefficient 5 (typically 3)
28:24	X	A_H_FILTER_P07C4: Phase 07 coefficient 4 (typically -14)
23:16	X	A_H_FILTER_P07C3: Phase 07 coefficient 3 (typically 67)
15:8	X	A_H_FILTER_P07C2: Phase 07 coefficient 2 (typically 85)
7:3	X	A_H_FILTER_P07C1: Phase 07 coefficient 1 (typically -16)
2:0	X	A_H_FILTER_P07C0: Phase 07 coefficient 0 (typically 3)

21.9.11 DC_WINC_A_H_FILTER_P08_0

Window A Horizontal Filter phase 08

Offset: 0x609 | Byte Offset: 0x1824 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P08C5: Phase 08 coefficient 5 (typically 3)
28:24	X	A_H_FILTER_P08C4: Phase 08 coefficient 4 (typically -15)
23:16	X	A_H_FILTER_P08C3: Phase 08 coefficient 3 (typically 76)
15:8	X	A_H_FILTER_P08C2: Phase 08 coefficient 2 (typically 76)
7:3	X	A_H_FILTER_P08C1: Phase 08 coefficient 1 (typically -15)
2:0	X	A_H_FILTER_P08C0: Phase 08 coefficient 0 (typically 3)

21.9.12 DC_WINC_A_H_FILTER_P09_0

Window A Horizontal Filter phase 09

Offset: 0x60a | Byte Offset: 0x1828 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P09C5: Phase 09 coefficient 5 (typically 3)
28:24	X	A_H_FILTER_P09C4: Phase 09 coefficient 4 (typically -16)
23:16	X	A_H_FILTER_P09C3: Phase 09 coefficient 3 (typically 85)
15:8	X	A_H_FILTER_P09C2: Phase 09 coefficient 2 (typically 67)
7:3	X	A_H_FILTER_P09C1: Phase 09 coefficient 1 (typically -14)
2:0	X	A_H_FILTER_P09C0: Phase 09 coefficient 0 (typically 3)

21.9.13 DC_WINC_A_H_FILTER_P0A_0

Window A Horizontal Filter phase 0A

Offset: 0x60b │ Byte Offset: 0x182c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P0AC5: Phase 0A coefficient 5 (typically 3)
28:24	X	A_H_FILTER_P0AC4: Phase 0A coefficient 4 (typically -15)
23:16	X	A_H_FILTER_P0AC3: Phase 0A coefficient 3 (typically 94)
15:8	X	A_H_FILTER_P0AC2: Phase 0A coefficient 2 (typically 56)
7:3	X	A_H_FILTER_P0AC1: Phase 0A coefficient 1 (typically -13)
2:0	X	A_H_FILTER_P0AC0: Phase 0A coefficient 0 (typically 3)

21.9.14 DC_WINC_A_H_FILTER_P0B_0

Window A Horizontal Filter phase 0B

Offset: 0x60c │ Byte Offset: 0x1830 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P0BC5: Phase 0B coefficient 5 (typically 3)
28:24	X	A_H_FILTER_P0BC4: Phase 0B coefficient 4 (typically -15)
23:16	X	A_H_FILTER_P0BC3: Phase 0B coefficient 3 (typically 102)
15:8	X	A_H_FILTER_P0BC2: Phase 0B coefficient 2 (typically 47)
7:3	X	A_H_FILTER_P0BC1: Phase 0B coefficient 1 (typically -11)
2:0	X	A_H_FILTER_P0BC0: Phase 0B coefficient 0 (typically 2)

21.9.15 DC_WINC_A_H_FILTER_P0C_0

Window A Horizontal Filter phase 0C

Offset: 0x60d │ Byte Offset: 0x1834 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P0CC5: Phase 0C coefficient 5 (typically 2)
28:24	X	A_H_FILTER_P0CC4: Phase 0C coefficient 4 (typically -13)
23:16	X	A_H_FILTER_P0CC3: Phase 0C coefficient 3 (typically 109)
15:8	X	A_H_FILTER_P0CC2: Phase 0C coefficient 2 (typically 37)
7:3	X	A_H_FILTER_P0CC1: Phase 0C coefficient 1 (typically -9)
2:0	X	A_H_FILTER_P0CC0: Phase 0C coefficient 0 (typically 2)

21.9.16 DC_WINC_A_H_FILTER_P0D_0

Window A Horizontal Filter phase 0D

Offset: 0x60e | Byte Offset: 0x1838 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P0DC5: Phase 0D coefficient 5 (typically 2)
28:24	X	A_H_FILTER_P0DC4: Phase 0D coefficient 4 (typically -11)
23:16	X	A_H_FILTER_P0DC3: Phase 0D coefficient 3 (typically 115)
15:8	X	A_H_FILTER_P0DC2: Phase 0D coefficient 2 (typically 27)
7:3	X	A_H_FILTER_P0DC1: Phase 0D coefficient 1 (typically -7)
2:0	X	A_H_FILTER_P0DC0: Phase 0D coefficient 0 (typically 2)

21.9.17 DC_WINC_A_H_FILTER_P0E_0

Window A Horizontal Filter phase 0E

Offset: 0x60f | Byte Offset: 0x183c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P0EC5: Phase 0E coefficient 5 (typically 1)
28:24	X	A_H_FILTER_P0EC4: Phase 0E coefficient 4 (typically -8)
23:16	X	A_H_FILTER_P0EC3: Phase 0E coefficient 3 (typically 122)
15:8	X	A_H_FILTER_P0EC2: Phase 0E coefficient 2 (typically 17)
7:3	X	A_H_FILTER_P0EC1: Phase 0E coefficient 1 (typically -5)
2:0	X	A_H_FILTER_P0EC0: Phase 0E coefficient 0 (typically 1)

21.9.18 DC_WINC_A_H_FILTER_P0F_0

Window A Horizontal Filter phase 0F

Offset: 0x610 | Byte Offset: 0x1840 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P0FC5: Phase 0F coefficient 5 (typically 1)
28:24	X	A_H_FILTER_P0FC4: Phase 0F coefficient 4 (typically -4)
23:16	X	A_H_FILTER_P0FC3: Phase 0F coefficient 3 (typically 124)
15:8	X	A_H_FILTER_P0FC2: Phase 0F coefficient 2 (typically 8)
7:3	X	A_H_FILTER_P0FC1: Phase 0F coefficient 1 (typically -2)
2:0	X	A_H_FILTER_P0FC0: Phase 0F coefficient 0 (typically 1)

21.9.19 DC_WINC_A_CSC_YOF_0

Color Space Conversion Coefficients

The CSC can be used for YUV to RGB conversion with brightness and hue/saturation control. The CSC can only be enabled for window A controlled by CSC_ENABLE register bits.

For Y color, the Y offset is applied first and saturation (clipping) is performed immediately after the Y offset is applied.

R = sat(KYRGB * sat(Y + YOF) + KUR * U + KVR * V)

G = sat(KYRGB * sat(Y + YOF) + KUG * U + KVG * V)

B = sat(KYRGB * sat(Y + YOF) + KUB * U + KVB * V)

Saturation and rounding is performed in the range of 0 to 255 for the above equations.

Typical values are:

YOF = -16.000, KYRGB = 1.1644

KUR = 0.0000, KVR = 1.5960

KUG = -0.3918, KVG = -0.8130

KUB = 2.0172, KVB = 0.0000

KUR and KVB are typically 0.0000 but they may be programmed non-zero for hue rotation.

The CSC can also take RGB input, in which case YOF, KVB, KUG, and KUR should be programmed to 0, and KYRGB will be forced to 0 by the hardware for generating R and B. KYRGB will not be forced to 0 for generating G. KVR, KYRGB, and KUB can be programmed to 1.0 or used as gain control for R, G, B, correspondingly.

Note that color value ranges from 0 to 255 for Y, R, G, B, and -128 to 127 for U and V.

Window A CSC Y Offset

Offset: 0x611 | Byte Offset: 0x1844 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_CSC_YOF: Y Offset in s.7.0 format

21.9.20 DC_WINC_A_CSC_KYRGB_0

Window A CSC Y Coefficient (gain) for RGB

Offset: 0x612 | Byte Offset: 0x1848 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	A_CSC_KYRGB: Y Gain for R, G, B colors in 2.8 format

21.9.21 DC_WINC_A_CSC_KUR_0

Window A CSC U coefficient for R

Offset: 0x613 | Byte Offset: 0x184c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	A_CSC_KUR: U coefficients for R in s.2.8 format

21.9.22 DC_WINC_A_CSC_KVR_0

Window A CSC V coefficient for R

Offset: 0x614 | Byte Offset: 0x1850 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	A_CSC_KVR: V coefficients for R in s.2.8 format

21.9.23 DC_WINC_A_CSC_KUG_0

Window A CSC U coefficient for G

Offset: 0x615 | Byte Offset: 0x1854 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	A_CSC_KUG: U coefficients for G in s.1.8 format

21.9.24 DC_WINC_A_CSC_KVG_0

Window A CSC V coefficient for G

Offset: 0x616 | Byte Offset: 0x1858 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	A_CSC_KVG: V coefficients for G in s.1.8 format

21.9.25 DC_WINC_A_CSC_KUB_0

Window A CSC U coefficient for B

Offset: 0x617 | Byte Offset: 0x185c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	A_CSC_KUB: U coefficients for B in s.2.8 format

21.9.26 DC_WINC_A_CSC_KVB_0

Window A CSC V coefficient for B

Offset: 0x618 | Byte Offset: 0x1860 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	A_CSC_KVB: V coefficients for B in s.2.8 format

21.9.27 DC_WINC_A_V_FILTER_P00_0

Vertical Scaling Filter Coefficients

The vertical scaling filter is a 2-tap filter with 4-bit positional phase. Coefficients 0 and 1 are 8-bit unsigned values ranging from 0 to 128.

Coefficient 0 is the multiplier for the earlier pixel (P0) in the group of 2 pixels, and coefficient 1 is the multiplier for the later pixel (P1) in the group. The output pixel positional phase is defined as centered in P0 if the positional phase is 0 or proportionally in between P0 and P1 if the positional phase is larger than 0.

The sum of all coefficients for each phase should be 128 typically. Therefore coefficient 1 can be calculated from (1 - coefficient 0) and only coefficient 0 is programmed. For each vertical positional phase, the filter coefficient requires 8 register bits. Note that color value ranges from 0 to 255 for Y, R, G, B, and -128 to 127 for U and V.

Window A Vertical Filter phase 00

Offset: 0x619 | Byte Offset: 0x1864 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P00C0: Phase 00 coefficient 0 (typically 128)

21.9.28 DC_WINC_A_V_FILTER_P01_0

Window A Vertical Filter phase 01

Offset: 0x61a | Byte Offset: 0x1868 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P01C0: Phase 01 coefficient 0 (typically 120)

21.9.29 DC_WINC_A_V_FILTER_P02_0

Window A Vertical Filter phase 02

Offset: 0x61b | Byte Offset: 0x186c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P02C0: Phase 02 coefficient 0 (typically 112)

21.9.30 DC_WINC_A_V_FILTER_P03_0

Window A Vertical Filter phase 03

Offset: 0x61c | Byte Offset: 0x1870 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P03C0: Phase 03 coefficient 0 (typically 104)

21.9.31 DC_WINC_A_V_FILTER_P04_0

Window A Vertical Filter phase 04

Offset: 0x61d | Byte Offset: 0x1874 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P04C0: Phase 04 coefficient 0 (typically 96)

21.9.32 DC_WINC_A_V_FILTER_P05_0

Window A Vertical Filter phase 05

Offset: 0x61e | Byte Offset: 0x1878 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P05C0: Phase 05 coefficient 0 (typically 88)

21.9.33 DC_WINC_A_V_FILTER_P06_0

Window A Vertical Filter phase 06

Offset: 0x61f | Byte Offset: 0x187c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P06C0: Phase 06 coefficient 0 (typically 80)

21.9.34 DC_WINC_A_V_FILTER_P07_0

Window A Vertical Filter phase 07

Offset: 0x620 | Byte Offset: 0x1880 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P07C0: Phase 07 coefficient 0 (typically 72)

21.9.35 DC_WINC_A_V_FILTER_P08_0

Window A Vertical Filter phase 08

Offset: 0x621 | Byte Offset: 0x1884 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P08C0: Phase 08 coefficient 0 (typically 64)

21.9.36 DC_WINC_A_V_FILTER_P09_0

Window A Vertical Filter phase 09

Offset: 0x622 | Byte Offset: 0x1888 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P09C0: Phase 09 coefficient 0 (typically 56)

21.9.37 DC_WINC_A_V_FILTER_P0A_0

Window A Vertical Filter phase 0A

Offset: 0x623 | Byte Offset: 0x188c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P0AC0: Phase 0A coefficient 0 (typically 48)

21.9.38 DC_WINC_A_V_FILTER_P0B_0

Window A Vertical Filter phase 0B

Offset: 0x624 | Byte Offset: 0x1890 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P0BC0: Phase 0B coefficient 0 (typically 40)

21.9.39 DC_WINC_A_V_FILTER_P0C_0

Window A Vertical Filter phase 0C

Offset: 0x625 | Byte Offset: 0x1894 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P0CC0: Phase 0C coefficient 0 (typically 32)

21.9.40 DC_WINC_A_V_FILTER_P0D_0

Window A Vertical Filter phase 0D

Offset: 0x626 | Byte Offset: 0x1898 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P0DC0: Phase 0D coefficient 0 (typically 24)

21.9.41 DC_WINC_A_V_FILTER_P0E_0

Window A Vertical Filter phase 0E

Offset: 0x627 | Byte Offset: 0x189c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P0EC0: Phase 0E coefficient 0 (typically 16)

21.9.42 DC_WINC_A_V_FILTER_P0F_0

Window A Vertical Filter phase 0F

Offset: 0x628 | Byte Offset: 0x18a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P0FC0: Phase 0F coefficient 0 (typically 8)

21.9.43 DC_WINC_A_H_FILTER_HI_P00_0

Window A Horizontal Filter phase 00

Offset: 0x629 | Byte Offset: 0x18a4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	A_H_FILTER_HI_P00C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

Bit	Reset	Description
7:6	X	A_H_FILTER_HI_P00C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	A_H_FILTER_HI_P00C3: MSB 9:9 of the lobe 3.
4	X	A_H_FILTER_HI_P00C2: MSB 9:9 of the lobe 2.
3:2	X	A_H_FILTER_HI_P00C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	A_H_FILTER_HI_P00C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.9.44 DC_WINC_A_H_FILTER_HI_P01_0

Window A Horizontal Filter phase 01

Offset: 0x62a | Byte Offset: 0x18a8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	A_H_FILTER_HI_P01C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	A_H_FILTER_HI_P01C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	A_H_FILTER_HI_P01C3: MSB 9:9 of the lobe 3.
4	X	A_H_FILTER_HI_P01C2: MSB 9:9 of the lobe 2.
3:2	X	A_H_FILTER_HI_P01C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	A_H_FILTER_HI_P01C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.9.45 DC_WINC_A_H_FILTER_HI_P02_0

Window A Horizontal Filter phase 02

Offset: 0x62b | Byte Offset: 0x18ac | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	A_H_FILTER_HI_P02C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	A_H_FILTER_HI_P02C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	A_H_FILTER_HI_P02C3: MSB 9:9 of the lobe 3.
4	X	A_H_FILTER_HI_P02C2: MSB 9:9 of the lobe 2.
3:2	X	A_H_FILTER_HI_P02C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	A_H_FILTER_HI_P02C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.9.46 DC_WINC_A_H_FILTER_HI_P03_0

Window A Horizontal Filter phase 03

Offset: 0x62c | Byte Offset: 0x18b0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	A_H_FILTER_HI_P03C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	A_H_FILTER_HI_P03C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	A_H_FILTER_HI_P03C3: MSB 9:9 of the lobe 3.
4	X	A_H_FILTER_HI_P03C2: MSB 9:9 of the lobe 2.
3:2	X	A_H_FILTER_HI_P03C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	A_H_FILTER_HI_P03C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.9.47 DC_WINC_A_H_FILTER_HI_P04_0

Window A Horizontal Filter phase 04

Offset: 0x62d | Byte Offset: 0x18b4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	A_H_FILTER_HI_P04C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	A_H_FILTER_HI_P04C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	A_H_FILTER_HI_P04C3: MSB 9:9 of the lobe 3.
4	X	A_H_FILTER_HI_P04C2: MSB 9:9 of the lobe 2.
3:2	X	A_H_FILTER_HI_P04C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	A_H_FILTER_HI_P04C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.9.48 DC_WINC_A_H_FILTER_HI_P05_0

Window A Horizontal Filter phase 05

Offset: 0x62e | Byte Offset: 0x18b8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	A_H_FILTER_HI_P05C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	A_H_FILTER_HI_P05C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	A_H_FILTER_HI_P05C3: MSB 9:9 of the lobe 3.
4	X	A_H_FILTER_HI_P05C2: MSB 9:9 of the lobe 2.
3:2	X	A_H_FILTER_HI_P05C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.

Bit	Reset	Description
1:0	X	A_H_FILTER_HI_P05C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.9.49 DC_WINC_A_H_FILTER_HI_P06_0

Window A Horizontal Filter phase 06

Offset: 0x62f | Byte Offset: 0x18bc | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	A_H_FILTER_HI_P06C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	A_H_FILTER_HI_P06C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	A_H_FILTER_HI_P06C3: MSB 9:9 of the lobe 3.
4	X	A_H_FILTER_HI_P06C2: MSB 9:9 of the lobe 2.
3:2	X	A_H_FILTER_HI_P06C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	A_H_FILTER_HI_P06C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.9.50 DC_WINC_A_H_FILTER_HI_P07_0

Window A Horizontal Filter phase 07

Offset: 0x630 | Byte Offset: 0x18c0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	A_H_FILTER_HI_P07C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	A_H_FILTER_HI_P07C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	A_H_FILTER_HI_P07C3: MSB 9:9 of the lobe 3.
4	X	A_H_FILTER_HI_P07C2: MSB 9:9 of the lobe 2.
3:2	X	A_H_FILTER_HI_P07C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	A_H_FILTER_HI_P07C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.9.51 DC_WINC_A_H_FILTER_HI_P08_0

Window A Horizontal Filter phase 08

Offset: 0x631 | Byte Offset: 0x18c4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	A_H_FILTER_HI_P08C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	A_H_FILTER_HI_P08C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.

Bit	Reset	Description
5	X	A_H_FILTER_HI_P08C3: MSB 9:9 of the lobe 3.
4	X	A_H_FILTER_HI_P08C2: MSB 9:9 of the lobe 2.
3:2	X	A_H_FILTER_HI_P08C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	A_H_FILTER_HI_P08C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.9.52 DC_WINC_A_H_FILTER_HI_P09_0

Window A Horizontal Filter phase 09

Offset: 0x632 | Byte Offset: 0x18c8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	A_H_FILTER_HI_P09C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	A_H_FILTER_HI_P09C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	A_H_FILTER_HI_P09C3: MSB 9:9 of the lobe 3.
4	X	A_H_FILTER_HI_P09C2: MSB 9:9 of the lobe 2.
3:2	X	A_H_FILTER_HI_P09C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	A_H_FILTER_HI_P09C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.9.53 DC_WINC_A_H_FILTER_HI_P0A_0

Window A Horizontal Filter phase 0A

Offset: 0x633 | Byte Offset: 0x18cc | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	A_H_FILTER_HI_P0AC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	A_H_FILTER_HI_P0AC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	A_H_FILTER_HI_P0AC3: MSB 9:9 of the lobe 3.
4	X	A_H_FILTER_HI_P0AC2: MSB 9:9 of the lobe 2.
3:2	X	A_H_FILTER_HI_P0AC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	A_H_FILTER_HI_P0AC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.9.54 DC_WINC_A_H_FILTER_HI_P0B_0

Window A Horizontal Filter phase 0B

Offset: 0x634 | Byte Offset: 0x18d0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	A_H_FILTER_HI_P0BC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	A_H_FILTER_HI_P0BC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	A_H_FILTER_HI_P0BC3: MSB 9:9 of the lobe 3.
4	X	A_H_FILTER_HI_P0BC2: MSB 9:9 of the lobe 2.
3:2	X	A_H_FILTER_HI_P0BC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	A_H_FILTER_HI_P0BC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.9.55 DC_WINC_A_H_FILTER_HI_P0C_0

Window A Horizontal Filter phase 0C

Offset: 0x635 | Byte Offset: 0x18d4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	A_H_FILTER_HI_P0CC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	A_H_FILTER_HI_P0CC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	A_H_FILTER_HI_P0CC3: MSB 9:9 of the lobe 3.
4	X	A_H_FILTER_HI_P0CC2: MSB 9:9 of the lobe 2.
3:2	X	A_H_FILTER_HI_P0CC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	A_H_FILTER_HI_P0CC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.9.56 DC_WINC_A_H_FILTER_HI_P0D_0

Window A Horizontal Filter phase 0D

Offset: 0x636 | Byte Offset: 0x18d8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	A_H_FILTER_HI_P0DC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	A_H_FILTER_HI_P0DC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	A_H_FILTER_HI_P0DC3: MSB 9:9 of the lobe 3.
4	X	A_H_FILTER_HI_P0DC2: MSB 9:9 of the lobe 2.
3:2	X	A_H_FILTER_HI_P0DC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.

Bit	Reset	Description
1:0	X	A_H_FILTER_HI_P0DC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.9.57 DC_WINC_A_H_FILTER_HI_P0E_0

Window A Horizontal Filter phase 0E

Offset: 0x637 | Byte Offset: 0x18dc | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	A_H_FILTER_HI_P0EC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	A_H_FILTER_HI_P0EC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	A_H_FILTER_HI_P0EC3: MSB 9:9 of the lobe 3.
4	X	A_H_FILTER_HI_P0EC2: MSB 9:9 of the lobe 2.
3:2	X	A_H_FILTER_HI_P0EC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	A_H_FILTER_HI_P0EC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.9.58 DC_WINC_A_H_FILTER_HI_P0F_0

Window A Horizontal Filter phase 0F

Offset: 0x638 | Byte Offset: 0x18e0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	A_H_FILTER_HI_P0FC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	A_H_FILTER_HI_P0FC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	A_H_FILTER_HI_P0FC3: MSB 9:9 of the lobe 3.
4	X	A_H_FILTER_HI_P0FC2: MSB 9:9 of the lobe 2.
3:2	X	A_H_FILTER_HI_P0FC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	A_H_FILTER_HI_P0FC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

The registers under DC_WIN are double buffered.

21.9.59 DC_WIN_A_WIN_OPTIONS_0

Window A Options

Class: Display Window Settings

Display Window A parameters

Offset: 0x700 │ Byte Offset: 0x1c00 │ Read/Write: R/W │ Reset: 0x00XXXXXX (0b00xxxxxx0xxxxxx0x0xxxxxxxxx0xxxx)

Bit	Reset	Description
31	0x0	A_H_FILTER_MODE: Horizontal filter mode. 0 = With previous Tegra horizontal coeffients widths. 0,5 - signed 5 bits 1,4 - signed 9 bits 2,3 - unsigned 9 bits 1 = New mode. With the expanded horizontal filter coefficients widths. 0,5- signed 5 bits 1,4- signed 7 bits 2,3- unsigned 9 bits 0 = OLD 1 = NEW
30	0x0	A_WIN_ENABLE: Window A Window enable 0 = DISABLE 1 = ENABLE
23	0x0	A_INTERLACE_ENABLE: Window A Interlace enable. This controls the fetch unit toggle between odd and even (normal) base addresses at the start of each field. Base address select between odd/even field is determined by HVTGEN field status. 0 = DISABLE 1 = ENABLE
22	X	A_YUV_RANGE_EXPAND: Window A Enable range expansion in the cases where RANGEREDFRM is 1 from mpd. Formula: Y = clip((Y-128)*2 + 128); Cb = clip((Cb-128)*2 + 128); Cr = clip((Cr-128)*2 + 128); where clip() function clips between 0 and 255. 0 = DISABLE 1 = ENABLE
20	X	A_DV_ENABLE: Window A Digital Vibrance Enable 0 = DISABLE 1 = ENABLE
18	X	A_CSC_ENABLE: Window A Color Space Conversion Enable. This controls the color space conversion and should be enabled for YCbCr/YUV color modes for conversion to B8G8R8 and for hue and saturation control. This can also be used for gain control for RGB color modes. 0 = DISABLE 1 = ENABLE
16	0x0	A_CP_ENABLE: Window A Color Palette Enable. This controls the color palette and should be enabled for palletized color modes. For non-palletized color modes, the color palette can be enabled for gamma correction. 0 = DISABLE 1 = ENABLE
14	0x0	A_V_FILTER_UV_ALIGN: Window A V Filter UV Alignment. This is effective only when vertical scaling filter is enabled and only on these formats YCbCr420P, YUV420P, YCbCr422R, YUV422R, YCbCr422RA YUV422RA. When UV alignment is enabled, the chroma components are aligned to the even number of luma component lines. When disabled the chroma components are aligned to half a pixel below the corresponding even number of luma component lines. It is usually disabled unless the incoming video stream specifically indicates otherwise. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
12	X	A_V_FILTER_OPTIMIZE: Window A V Filter Optimization. This is effective only when vertical scaling filter is enabled. This can be used (enabled) to temporarily disable the vertical scaling filter when the vertical scaling DDA fraction is zero. In this case the next line is not fetched from memory to save bandwidth and power. This feature cannot be used in 420P/422R/422RA formats. 0 = DISABLE 1 = ENABLE
10	X	A_V_FILTER_ENABLE: Window A V Filter Enable. This controls V scaling filter and is effective only for non-palletized color modes. If V filter is disabled, only one line is read from memory for each output line. NOTE: This feature is not supported,. This bit is ignored and is a don't care. 0 = DISABLE 1 = ENABLE
8	X	A_H_FILTER_ENABLE: Window A H Filter Enable. This controls H scaling filter and is effective only for non-palletized color modes. 0 = DISABLE 1 = ENABLE
6	X	A_COLOR_EXPAND: Window A 12/15/16/18-to-24 bpp color expansion. This bit should be enabled only for 12-bpp B4G4R4A4, 15-bpp B5G5R5A, 16-bpp B5G6R5, 18-bpp B6G6R6 color modes. If enabled the color conversion is performed prior to horizontal scaling. 0 = DISABLE 1 = ENABLE
4	0x0	A_SCAN_COLUMN: Window A Scanning direction 0= Scan in horizontal direction (for 0 or 180 degrees) 1= Scan in vertical direction (for 90 or 270 degrees) 0 = DISABLE 1 = ENABLE
2	X	A_V_DIRECTION: Window A Vertical (Y) drawing Direction 0 = INCREMENT 1 = DECREMENT
0	X	A_H_DIRECTION: Window A Horizontal (X) drawing Direction 0 = INCREMENT 1 = DECREMENT

21.9.60 DC_WIN_A_BYTE_SWAP_0

Window A Byte Swap

Offset: 0x701 | Byte Offset: 0x1c04 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2:0	X	A_BYTE_SWAP: Window A Byte Swap. This controls byte swap of frame data read from memory prior to any data processing in the display module. 00= no byte swap (3 2 1 0) 001= byte swap for each 2-byte word (2 3 0 1) 010= byte swap for each 4-byte word (0 1 2 3) 011= word swap for each 4-byte word (1 0 3 2) 100= byte0 swapped with byte2 (3 0 1 2) 101= left shift every byte (2 1 0 3) 0 = NOSWAP 1 = SWAP2 2 = SWAP4 3 = SWAP4HW 4 = SWAP02 5 = SWAPLEFT

21.9.61 DC_WIN_A_COLOR_DEPTH_0

Window A Color Depth For YCbCr data format, Cb and Cr are 8-bit unsigned values. For YUV data format, U and V are 8-bit signed values.

YCbCr422R is similar to YCbCr422P but the Cb and Cr are shared vertically.

YUV422R is similar to YUV422P but the U and V are shared vertically.

YCbCr422RA is the same as YCbCr422R in memory, and YUV422RA is the same as YUV422R in memory. However, while reading from memory, for YCbCr422RA and YUV422RA, chroma averaging is applied for each pixel pair so that they can be processed as YUV422 by the display pipeline.

For YCbCr422R and YUV422R, every other chroma pixels are not used (discarded) by the display pipeline. B6x2G6x2R6x2A8 is similar to B8G8R6A8 but with the 2 LSBs zeroed out.

R6x2G6x2B6x2A8 is similar to R8G8B6A8 but with the 2 LSBs zeroed out. Some formats have NVCF_ or T_ prefix aliases for NVidia surface format naming convention, wherein left-to-right component names correspond to MSB to LSB bits within a word. In old names without NVCF_ or T_, left-to-right component names generally map LSB to MSB within a word.

Offset: 0x703 | Byte Offset: 0x1c0c | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0001100)

Bit	Reset	Description
6:0	B8G8R8A8	A_COLOR_DEPTH: Window A Color Depth. Supported color depths are: P8 = 8-bpp (palletized) B4G4R4A4 = 12-bpp B4G4R4 B5G5R5A = 15-bpp B5G5R5 AB5G5R5 = 15-bpp B5G5R5 B5G6R5 = 16-bpp B5G6R5 B8G8R8A8 = 32-bpp B8G8R8A8 R8G8B8A8 = 32-bpp R8G8B8A8 YCbCr422 = 16-bpp YCbCr422 packed YUV422 = 16-bpp YUV422 YCbCr420P = 16-bpp YCbCr420 planar YUV420P = 16-bpp YUV420 planar YCbCr422P = 16-bpp YCbCr422 planar YUV422P = 16-bpp YUV422 planar YCbCr422R = 16-bpp YCbCr422 rotated planar YUV422R = 16-bpp YUV422 rotated planar YCbCr422RA= 16-bpp YCbCr422 rotated planar with chroma averaging YUV422RA = 16-bpp YUV422 rotated planar with chroma averaging. Reserved values: 0,1,2,14,15,24,25. 3 = T_P8, P8 4 = T_A4R4G4B4, B4G4R4A4 5 = T_A1R5G5B5, B5G5R5A 6 = T_R5G6B5, B5G6R5 7 = T_R5G5B5A1, AB5G5R5 12 = T_A8R8G8B8, B8G8R8A8 13 = T_A8B8G8R8, R8G8B8A8

Bit	Reset	Description
		16 = T_U8_Y8__V8_Y8, CbYCrY422, YCbCr422 17 = T_U8_Y8__V8_Y8_TRUE, UYVY422, YUV422, TRUE_U8_Y8__V8_Y8 18 = T_Y8___U8___V8_N420, YCbCr420P 19 = T_Y8___U8___V8_N420_TRUE, YUV420P 20 = T_Y8___U8___V8_N422, YCbCr422P 21 = T_Y8___U8___V8_N422_TRUE, YUV422P 22 = T_Y8___U8___V8_N422R 22 = YCbCr422RP, YCbCr422R 23 = T_Y8___U8___V8_N422R_TRUE 23 = YUV422RP, YUV422R 24 = T_V8_Y8__U8_Y8, CrYCbY422 25 = T_V8_Y8__U8_Y8_TRUE, VYUY422 26 = T_Y8, Y8 27 = T_A4B4G4R4, R4G4B4A4 28 = T_A1B5G5R5, R5G5B5A 29 = T_B5G5R5A1, AR5G5B5 30 = T_X1R5G5B5, B5G5R5X1 31 = T_R5G5B5X1, X1B5G5R5 32 = T_X1B5G5R5, R5G5B5X1 33 = T_B5G5R5X1, X1R5G5B5 34 = T_B5G6R5, R5G6B5 35 = T_B8G8R8A8 36 = T_R8G8B8A8 37 = T_X8R8G8B8, B8G8R8X8 38 = T_X8B8G8R8, R8G8B8X8 39 = T_A8Y8U8V8 40 = T_V8U8Y8A8 41 = T_Y8___U8___V8_N444, YCbCr444P 42 = T_Y8___U8V8_N420, YCrCb420SP 43 = T_Y8___V8U8_N420, YCbCr420SP 44 = T_Y8___U8V8_N422, YCrCb422SP 45 = T_Y8___V8U8_N422, YCbCr422SP 46 = T_Y8___U8V8_N422R, YCrCb422RSP 47 = T_Y8___V8U8_N422R, YCbCr422RSP 48 = T_Y8___U8V8_N444, YCrCb444SP 49 = T_Y8___V8U8_N444, YCbCr444SP 50 = T_A8Y8U8V8_TRUE 51 = T_V8U8Y8A8_TRUE 52 = T_Y8___U8___V8_N444_TRUE, YUV444P 53 = T_Y8___U8V8_N420_TRUE, YVU420SP 54 = T_Y8___V8U8_N420_TRUE, YUV420SP 55 = T_Y8___U8V8_N422_TRUE, YVU422SP 56 = T_Y8___V8U8_N422_TRUE, YUV422SP 57 = T_Y8___U8V8_N422R_TRUE, YVU422RSP 58 = T_Y8___V8U8_N422R_TRUE, YUV422RSP 59 = T_Y8___U8V8_N444_TRUE, YUV444SPYVU444SP 60 = T_Y8___V8U8_N444_TRUE, YVU444SPYUV444SP 61 = T_Y8_U8__Y8_V8, YCbYCr422 62 = T_Y8_U8__Y8_V8_TRUE, YUYV422 63 = T_Y8_V8__Y8_U8, YCrYCb422 64 = T_Y8_V8__Y8_U8_TRUE, YVYU422 65 = T_R8G8B8X8 66 = T_B8G8R8X8

21.9.62 DC_WIN_A_POSITION_0

Window A Position

This register defines the H position and size of Window A after scaling (if there is any)

Offset: 0x704 │ Byte Offset: 0x1c10 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	A_V_POSITION: Window A V Position. This is specified with respect to the top edge of active display area.
12:0	X	A_H_POSITION: Window A H Position. This is specified with respect to the left edge of active display area.

21.9.63 DC_WIN_A_SIZE_0

Window A Size

This register defines the V position and size of Window A after scaling (if there is any)

Note: programming on window size should guarantee the whole window is inside the active area, otherwise extra rows/columns will be fetched and discarded which affects performance.

Offset: 0x705 │ Byte Offset: 0x1c14 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	A_V_SIZE: Window A V Size (lines). This is the vertical size after scaling.
12:0	X	A_H_SIZE: Window A H Size (pixels). This is the horizontal size after scaling.

21.9.64 DC_WIN_A_PRESCALED_SIZE_0

Window A Pre-scaled Size

This register defines Window A pre-scaled size.

The H pre-scaled size is needed to determine how many bytes to fetch from memory per line and this parameter must be programmed exactly as needed taking into account the scaling factor. For planar YUV or YCbCr data formats, this parameter refer to the H pre-scaled of the Y plane.

The total number of lines to be fetched from memory is determined by post-scale V size but V pre-scaled size is needed to 'clamp' the last valid line if the vertical DDA is exactly or slightly beyond the specified V pre-scaled size.

H pre-scaled size ideally should be in terms of pixel but then hardware needs to convert this precisely to bytes to determine the amount of data to request from memory.

Offset: 0x706 │ Byte Offset: 0x1c18 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	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.

21.9.65 DC_WIN_A_H_INITIAL_DDA_0

Window A H Initial DDA

The first pixel of pre-scaled image is always used to output the first pixel so essentially this is the same as forcing the H Initial DDA integer portion to 1 initially even though the user typically programs this to 0. If it makes the implementation easier, it is possible to force software to program the Initial DDA integer portion to 1. Similarly, with the V Initial DDA.

Offset: 0x707 | Byte Offset: 0x1c1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	A_H_INITIAL_DDA: Window A H Initial DDA (4.12). This is typically programmed to 0.0

21.9.66 DC_WIN_A_V_INITIAL_DDA_0

Window A V Initial DDA

Offset: 0x708 | Byte Offset: 0x1c20 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	A_V_INITIAL_DDA: Window A V Initial DDA (4.12). This is typically programmed to 0.0 for both non-interlaced and interlaced sources.

21.9.67 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)) + initial_dda_bias, MAX)
- Filter off: min(round(prescaled_size_in_pixels * 0x1000 / (post_scaled_size_in_pixels - 1) - 0.5) + initial_dda_bias, 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/pixel formats.

8.0 (0x8000) for 2 Bytes/pixel formats.

Where the value of initial_dda_bias is as follows:

For V_DDA_INCREMENT: initial_dda_bias = (v_intial_dda == negative) ? mod(v_initial_dda)/(post_scaled_size_in_pixels - 1) : 0 ;

For H_DDA_INCREMENT: initial_dda_bias = 0;

They are theoretically the biggest values that guarantee not displaying beyond an image boundary. If the DDA increment is less than 1.0, then the image is upscaled. If the DDA increment is more than 1.0, then the image is downscaled.

Offset: 0x709 | Byte Offset: 0x1c24 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	A_V_DDA_INCREMENT: Window A Vertical DDA Increment (4.12). This should be set to 1.0 if there is no scaling. Maximum value is 15.0 regardless of the number of bytes per pixel.

Bit	Reset	Description
15:0	X	A_H_DDA_INCREMENT: Window A Horizontal DDA Increment (4.12). This should be set to 1.0 if there is no scaling. The maximum value for downscaling depends on the number of bytes per pixel. For 4-byte/pixel modes (32-bpp) the maximum value is 4.0 and for all other modes the maximum value is 8.0.

21.9.68 DC_WIN_A_LINE_STRIDE_0

Window A Line Stride

Offset: 0x70a │ Byte Offset: 0x1c28 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	A_UV_LINE_STRIDE: Window A Line Stride for Chroma. This is stride (in bytes) for planar YUV or YCbCr data formats for the chroma plane, with the restriction that it must be programmed to be multiples of 4 (16 if tiled or in horizontal flipping) This is not used (ignored) for other non-planar data formats.
15:0	X	A_LINE_STRIDE: Window A Line Stride. This is stride (in bytes) for all non-planar data formats. If the memory surface is tiled, the stride needs to be a multiple of 16. If H_DIRECTION of window A is set to DECREMENT, the stride also needs to be a multiple of 16. For planar YUV or YCbCr data formats, this is stride (in bytes) for the luma plane with the restriction that it must be multiples of 8 (16 if tiled or in horizontal flipping) For tiled surface this value may affect starting address of a window. Refer to the comment of START_ADDR for detail.

21.9.69 DC_WIN_A_DV_CONTROL_0

Window A Digital Vibrance Control

If enabled, Digital Vibrance is applied after H and V scaling and after color palette or color space conversion logic but before color keying multiplexer and before cursor multiplexer.

- After DV, new R = R + (2R - G - B) * FR, where FR is fraction from 0 to 7/8
- After DV, new G = G + (2G - R - B) * FG, where FG is fraction from 0 to 7/8
- After DV, new B = B + (2B - R - G) * FB, where FB is fraction from 0 to 7/8

Offset: 0x70e │ Byte Offset: 0x1c38 │ Read/Write: R/W │ Reset: 0x000X0X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18:16	X	A_DV_CONTROL_B: Digital Vibrance control for B
10:8	X	A_DV_CONTROL_G: Digital Vibrance control for G
2:0	X	A_DV_CONTROL_R: Digital Vibrance control for R

Class: Display Color Keying and Blending

Color keying and blending of the display windows are done prior to cursor blending.

Cursor always goes on top of the blended windows. Blending is controlled independently on each possible overlap area of the display windows. If 3 windows are enabled there are 7 possible overlap area combinations. For every window in each overlap area combination, color key can be disabled or enabled. Also, for every window in each overlap area combination, there is a corresponding window blend control parameter and a window blend weight parameter. The window blend control parameter is

always effective but the window blend weight is not always used. The window blend weight can also be derived from pixel alpha value or from the reverse of other overlapping windows weight.

Color keying has the highest priority for display window blending. Color key consists of a range of color which is searched independently for each window. If more than 1 windows color key is enabled, then Window A color key has the highest priority, followed by Window B color key, and then followed by Window C color key. Two sets of color key ranges (Color Key 0 and Color Key 1) can be defined; they are shared for all windows. It is possible to use both color key sets for the same window or for two separate windows.

The two sets of color key ranges should not overlap. If they do, the overlapped colors are treated as if they are part of Color Key 0 and not part of Color Key 1.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is enabled for any overlap condition and the window pixel is not within the color key range (key not match), then the window pixel will not be blended with other overlapping window pixels but it will be weighted. The weight is not dependent on the overlap condition it is controlled by the same set of parameters for all overlapping condition.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is enabled for a particular overlap condition and the window pixel is within the color key range (key match), then the window pixel will be weighted and blended with other overlapping pixels and this is controlled separately for each overlap condition.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is disabled then the window pixel will be blended with other overlapping pixels and this is controlled separately for each overlap condition.

Display Color Key parameters

For B4G4R4A4, B5G6R5A, and B5G6R5 modes, the color key should be compared prior to color conversion to 24-bpp and unused least significant bits of the pixel are filled with zeros.

For palletized mode, the color key is compared prior to the color palette, and the palletized color is compared against the green color key values/mask.

For YUV mode, U and V are offset by +128 before performing the color key comparison. In all cases, the color key is compared prior to the horizontal/vertical scaling filter and prior to digital vibrance control.

Both upper and lower values are inclusive.

21.9.70 DC_WIN_A_BLEND_LAYER_CONTROL_0

Window A

Offset: 0x716 | Byte Offset: 0x1c58 | Read/Write: R/W | Reset: 0x01000000 (0bxxxx00010000000000000000000000)

Bit	Reset	Description
27:25	0x0	A_COLOR_KEY_SELECT: 0 = NONE 1 = WINDOWA_KEY0 2 = WINDOWA_KEY1 3 = WINDOWB_KEY0 4 = WINDOWB_KEY1 5 = WINDOWC_KEY0 6 = WINDOWC_KEY1
24	BLEND_BYPASS	A_BLEND_BYPASS: 0 = BLEND_ENABLE 1 = BLEND_BYPASS
23:16	0x0	A_K2

Bit	Reset	Description
15:8	0x0	A_K1
7:0	0x0	A_WINDOW_LAYER_DEPTH

21.9.71 DC_WIN_A_BLEND_MATCH_SELECT_0

Offset: 0x717 | Byte Offset: 0x1c5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xx00x000x000)

Bit	Reset	Description
13:12	0x0	A_BLEND_FACTOR_DST_ALPHA_MATCH_SELECT: 0 = ZERO 1 = ONE 2 = NEG_K1_TIMES_SRC 3 = K2
9:8	0x0	A_BLEND_FACTOR_SRC_ALPHA_MATCH_SELECT: 0 = ZERO 1 = K1 2 = K2
6:4	0x0	A_BLEND_FACTOR_DST_COLOR_MATCH_SELECT: 0 = ZERO 1 = ONE 2 = K1 3 = K2 4 = K1_TIMES_DST 5 = NEG_K1_TIMES_DST 6 = NEG_K1_TIMES_SRC 7 = NEG_K1
2:0	0x0	A_BLEND_FACTOR_SRC_COLOR_MATCH_SELECT: 0 = ZERO 1 = ONE 2 = K1 3 = K1_TIMES_DST 4 = NEG_K1_TIMES_DST 5 = K1_TIMES_SRC

21.9.72 DC_WIN_A_BLEND_NOMATCH_SELECT_0

Offset: 0x718 | Byte Offset: 0x1c60 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xx00x000x000)

Bit	Reset	Description
13:12	0x0	A_BLEND_FACTOR_DST_ALPHA_NOMATCH_SELECT: 0 = ZERO 1 = ONE 2 = NEG_K1_TIMES_SRC 3 = K2
9:8	0x0	A_BLEND_FACTOR_SRC_ALPHA_NOMATCH_SELECT: 0 = ZERO 1 = K1 2 = K2
6:4	0x0	A_BLEND_FACTOR_DST_COLOR_NOMATCH_SELECT: 0 = ZERO 1 = ONE 2 = K1 3 = K2 4 = K1_TIMES_DST 5 = NEG_K1_TIMES_DST 6 = NEG_K1_TIMES_SRC 7 = NEG_K1
2:0	0x0	A_BLEND_FACTOR_SRC_COLOR_NOMATCH_SELECT:

Bit	Reset	Description
		0 = ZERO 1 = ONE 2 = K1 3 = K1_TIMES_DST 4 = NEG_K1_TIMES_DST 5 = K1_TIMES_SRC

21.9.73 DC_WIN_A_BLEND_ALPHA_1BIT_0

Offset: 0x719 | Byte Offset: 0x1c64 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	A_BLEND_WEIGHT1: This is used only for 1-bit alpha and alpha value of 1.
7:0	0x0	A_BLEND_WEIGHT0: This is used only for 1-bit alpha and alpha value of 0.

21.10 WINBUF_A Registers

The registers under DC_WINBUF are triple-buffered.

21.10.1 DC_WINBUF_A_START_ADDR_0

Window A Start Address

Overview

START_ADDR, BUF_STRIDE, LINE_STRIDE, ADDR_H_OFFSET, ADDR_V_OFFSET combined, specify the starting address of a window buffer in the memory surface. Generally, START_ADDR is programmed with the starting address of the memory surface. H/V_OFFSET specifies the address offsets of the pixel at the beginning of the window with respect to the starting address of the memory surface. (There is an exception to that when memory surface is not well aligned, see ”For linear address mode” in Programming Restrictions below.)

Note that "beginning of the window" here means the top-left corner of a window in normal mode, top-right corner in horizontal flipping, bottom-left corner in vertical flipping, and bottom-right in horizontal+vertical flipping.

Starting Address Calculation

These formulae are same for both tiled and linear modes. However, for linear mode, the addresses calculated are real physical addresses. For tiled mode, these addresses will be translated to the real physical addresses by the memory controller client before being used.

- When a window is host triggered, starting address of a window is calculated as follows by hardware:
  - non-YUV-planar modes:

    starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - YUV-planar modes:

    Y-starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    U-starting-address = START_ADDR_U + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    V-starting-address = START_ADDR_V + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by hardware.
- When a window is non-host triggered, starting address of a window buffer is calculated as below.

- non-YUV-planar mode:

  starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

- YUV-planar mode:

  Y-starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

  U-starting-address = START_ADDR_U + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

  V-starting-address = START_ADDR_V + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by hardware.

  buf_index is the index transmitted by the triggering module pointing to the first buffer of a frame.

Programming Restrictions

i. For tiled address mode:

Image surface can only aligned to multiples of 256, thus the following restrictions.

- START_ADDR, START_ADDR_U, START_ADDR_V need to be multiples of 256.
- BUF_STRIDE, UV_BUF_STRIDE need to be multiples of 256
- LINE_STRIDE, UV_LINE_STRIDE need to be multiples of 16
- ADDR_H_OFFSET needs to be even in YUV planar format, or a multiple of bytes per pixel in other formats. If H_DIRECTION=DECREMENT, however, it should point to last valid byte, which is an odd offset.
- ADDR_V_OFFSET needs to be a multiple of 2 in YUV planar format, unless H_DIRECTION=DECREMENT, but with no restrictions on other color formats.

ii. For linear address mode:

Image surface can be aligned 2, 4 or 8 bytes, depending on the color formats.

As an additional restriction for display, START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16.

When a surface is not aligned to 16 bytes, program START_ADDR with the memory surface address with its least 4 significant bits zeroed out and add these 4 address bits to the original H_OFFSET. (So the formulae in Starting Address Calculation above still hold)

- For all formats:
  - START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16.
- For 16-bpp formats,
  - (START_ADDR+H_OFFSET) need to be a multiple of 2.
- For 32-bpp formats,
  - (START_ADDR+H_OFFSTE) needs to be a multiple of 4.
- For YUV planar formats:
  - BUF_STRIDE, UV_BUF_STRIDE:

    BUF_STRIDE[2:1]=UV_BUF_STRIDE[1:0]

    or as a stricter constraint: BUF_STRIDE is a multiple of 8, UV_BUF_STRIDE is a multiple of 4.
  - LINE_STRIDE, UV_LINE_STRIDE:

LINE_STRIDE and UV_LINE_STRIDE need to be at least 16.

LINE_STRIDE needs to be a multiple of 8, UV_LINE_STRIDE needs to be a multiple of 4.

- ADDR_H_OFFSET: Needs to be even unless H_DIRECTION=DECREMENT, in which case should point to the last byte pixel, which is at an odd offset. .
- ADDR_V_OFFSET: Needs to be even unless V_DIRECTION=DECREMENT, in which case should point to the last valid line, which is at an odd offset. .

iii. Memory allocation for non-host triggered case:

When a window buffer is not controlled by host (software) then a frame may be stored in multiple buffers. In this case, the buffers must be contiguous in the memory because display will use the same luma or chroma line strides for all lines in the frame. Also buffer wraparound must not occur in the middle of the displayed part of the frame.

The controlling module will send frame start and frame end indicators (flags) to display module to indicate the beginning and end of frame. Buffer start address is latched when frame start flag is active but the actual buffer start address is not switched until frame end flag is active. In the case where one buffer corresponds to one frame then frame start and frame end flag are active every time a buffer index is sent.

Offset: 0x800 | Byte Offset: 0x2000 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_START_ADDR: Window A Start Address. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

21.10.2 DC_WINBUF_A_START_ADDR_NS_0

Window A Shadowed Start Address

Offset: 0x801 | Byte Offset: 0x2004 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_START_ADDR_NS: Window A Shadowed Start Address. This is ARM set shadow of Start Address.

21.10.3 DC_WINBUF_A_START_ADDR_U_0

Window A Start Address for U plane

Offset: 0x802 | Byte Offset: 0x2008 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_START_ADDR_U: Window A Start Address for U plane. This is a byte address.

21.10.4 DC_WINBUF_A_START_ADDR_U_NS_0

Window A Shadowed Start Address for U plane

Offset: 0x803 | Byte Offset: 0x200c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_START_ADDR_U_NS: Window A Shadowed Start Address for U plane. This is ARM set shadow register of U start address

21.10.5 DC_WINBUF_A_START_ADDR_V_0

Window A Start Address for V plane

Offset: 0x804 | Byte Offset: 0x2010 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_START_ADDR_V: Window A Start Address for V plane. This is a byte address.

21.10.6 DC_WINBUF_A_START_ADDR_V_NS_0

Window A Shadowed Start Address for V plane

Offset: 0x805 | Byte Offset: 0x2014 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_START_ADDR_V_NS: Window A Shadowed Start Address for V plane. This is ARM set shadow register of U start address

21.10.7 DC_WINBUF_A_ADDR_H_OFFSET_0

Window A Horizontal Address Offset

Offset: 0x806 | Byte Offset: 0x2018 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_ADDR_H_OFFSET: Window A Horizontal address offset. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies horizontal offset of Y plane. The horizontal offsets of U/V plane is derived by hardware.

21.10.8 DC_WINBUF_A_ADDR_H_OFFSET_NS_0

Window A Shadowed Horizontal Address Offset

Offset: 0x807 | Byte Offset: 0x201c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_ADDR_H_OFFSET_NS: Window A Shadowed Horizontal address offset. This is ARM set shadow of ADDR_H_OFFSET

21.10.9 DC_WINBUF_A_ADDR_V_OFFSET_0

Window A Vertical Address Offset

Offset: 0x808 | Byte Offset: 0x2020 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_ADDR_V_OFFSET: Window A Vertical address offset. This is a line number. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies vertical offset of Y plane. The vertical offsets of U/V plane is derived by hardware.

21.10.10 DC_WINBUF_A_ADDR_V_OFFSET_NS_0

Window A Shadowed Vertical Address Offset

Offset: 0x809 | Byte Offset: 0x2024 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_ADDR_V_OFFSET_NS: Window A Shadowed Vertical address offset. This is the ARM set shadow of ADDR_V_OFFSET

21.10.11 DC_WINBUF_A_UFLOW_STATUS_0

Window A FIFO Underflow Status Register

Offset: 0x80a | Byte Offset: 0x2028 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	COUNT_OFLOW: Flag bit that indicates that the underflow event counter has overflowed. There were too many events. If COUNT_OFLOW is set, UFLOW_COUNT is meaningless. Cleared on write.
23:0	X	UFLOW_COUNT: Underflow count. This field indicates the number of contiguous groups of output pixels for which there was no data in the FIFO. For example, if the valid from the FIFO is low for 10 consecutive cycles and then goes high, the counter will increment by one. Reset to zero on write.

21.10.12 DC_WINBUF_A_SURFACE_KIND_0

Window A Surface Kind

Offset: 0x80b | Byte Offset: 0x202c | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxxxxxxxxxxxxxxxxx001xx00)

Bit	Reset	Description
6:4	0x1	A_BLOCK_HEIGHT: Block Height: For block linear, height of block in GOBs 0 = HEIGHT_1 1 = HEIGHT_2 2 = HEIGHT_4 3 = HEIGHT_8 4 = HEIGHT_16 5 = HEIGHT_32
1:0	0x0	A_SURFACE_KIND: Surface Kind: Pitched, Tiled, or Block Linear 0 = PITCH 1 = TILED 2 = BL_16B2

21.10.13 DC_WINBUF_A_SURFACE_WEIGHT_0

Window A Surface Weights

Offset: 0x80c | Byte Offset: 0x2030 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
6:5	X	A_SURFACE_WEIGHT_V: Window A V Surface Weights. 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16

Bit	Reset	Description
		0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
4:3	X	A_SURFACE_WEIGHT_U: Window A U or UV Surface Weights. 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16 0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
2:1	X	A_SURFACE_WEIGHT_Y: Window A Y or packed Surface Weights. 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16 0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
0	0x0	A_SURFACE_WEIGHT_OVERRIDE: Weight Override 0 = DISABLE 1 = ENABLE

21.10.14 DC_WINBUF_A_START_ADDR_HI_0

Window A Higher 2 bits of Start Address

Makes the start address 34 bits by adding the higher 2 bits below.

Offset: 0x80d │ Byte Offset: 0x2034 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	A_START_ADDR_HI: Window A Start Address. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

21.10.15 DC_WINBUF_A_START_ADDR_HI_NS_0

Window A Shadowed Higher 2 bits of Start Address

Makes the start address 34 bits by adding the higher 2 bits below.

Offset: 0x80e │ Byte Offset: 0x2038 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	A_START_ADDR_HI_NS: Window A Shadowed Start Address. This is ARM set shadow of Start Address.

21.10.16 DC_WINBUF_A_START_ADDR_HI_U_0

Window A Higher 2 bits of Start Address for U Plane

Makes the start address 34 bits by adding the higher 2 bits below.

Offset: 0x80f │ Byte Offset: 0x203c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	A_START_ADDR_HI_U: Window A Start Address for U plane. This is a byte address.

21.10.17 DC_WINBUF_A_START_ADDR_HI_U_NS_0

Window A Shadowed Higher 2 bits of Start Address for U Plane

Makes the start address 34 bits by adding the higher 2 bits below.

Offset: 0x810 │ Byte Offset: 0x2040 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	A_START_ADDR_HI_U_NS: Window A Shadowed Higher 2 bits of Start Address for U plane. This is ARM set shadow register of U start address

21.10.18 DC_WINBUF_A_START_ADDR_HI_V_0

Window A Higher 2 bits of Start Address for V Plane

Makes the start address 34 bits by adding the higher 2 bits below.

Offset: 0x811 │ Byte Offset: 0x2044 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	A_START_ADDR_HI_V: Window A Higher 2 bits of Start Address for V plane. This is a byte address.

21.10.19 DC_WINBUF_A_START_ADDR_HI_V_NS_0

Window A Shadowed Higher 2 bits of Start Address for V Plane

Makes the start address 34 bits by adding the higher 2 bits below.

Offset: 0x812 │ Byte Offset: 0x2048 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	A_START_ADDR_HI_V_NS: Window A Shadowed Higher 2 bits of Start Address for V plane. This is ARM set shadow register of U start address

21.10.20 DC_WINBUF_A_START_ADDR_FIELD2_0

Window A Start Address

Interlace/Stereo support. 64-bit base address for the odd field/right frame.

Offset: 0x813 | Byte Offset: 0x204c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_START_ADDR_FIELD2: Window A Start Address. This is a byte address. The LSB is not used for the 16-bpp non-planar pixel format, and the last 2 LSBs are not used for the 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

21.10.21 DC_WINBUF_A_START_ADDR_FIELD2_NS_0

Window A Shadowed Start Address

Offset: 0x814 | Byte Offset: 0x2050 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_START_ADDR_FIELD2_NS: Window A Shadowed Start Address. This is the ARM set shadow of the start address.

21.10.22 DC_WINBUF_A_START_ADDR_FIELD2_U_0

Window A Start Address for U plane

Offset: 0x815 | Byte Offset: 0x2054 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_START_ADDR_FIELD2_U: Window A Start Address for U plane. This is a byte address.

21.10.23 DC_WINBUF_A_START_ADDR_FIELD2_U_NS_0

Window A Shadowed Start Address for U plane

Offset: 0x816 | Byte Offset: 0x2058 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_START_ADDR_FIELD2_U_NS: Window A Shadowed Start Address for U plane. This is the ARM set shadow register of the U start address.

21.10.24 DC_WINBUF_A_START_ADDR_FIELD2_V_0

Window A Start Address for V plane

Offset: 0x817 | Byte Offset: 0x205c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_START_ADDR_FIELD2_V: Window A Start Address for V plane. This is a byte address.

21.10.25 DC_WINBUF_A_START_ADDR_FIELD2_V_NS_0

Window A Shadowed Start Address for V plane

Offset: 0x818 | Byte Offset: 0x2060 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_START_ADDR_FIELD2_V_NS: Window A Shadowed Start Address for V plane. This is the ARM set shadow register of the U start address.

21.10.26 DC_WINBUF_A_START_ADDR_FIELD2_HI_0

Window A Higher 2 bits of Start Address

Makes the start address 34 bits by adding the higher 2 bits below.

Offset: 0x819 | Byte Offset: 0x2064 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	A_START_ADDR_FIELD2_HI: Window A Start Address. This is a byte address. The LSB is not used for the 16-bpp non-planar pixel format, and the last 2 LSBs are not used for the 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies the start address for the Y plane.

21.10.27 DC_WINBUF_A_START_ADDR_FIELD2_HI_NS_0

Window A Shadowed Higher 2 bits of Start Address

Makes the start address 34 bits by adding the higher 2 bits below.

Offset: 0x81a | Byte Offset: 0x2068 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	A_START_ADDR_FIELD2_HI_NS: Window A Shadowed Start Address. This is the ARM set shadow of the start address.

21.10.28 DC_WINBUF_A_START_ADDR_FIELD2_HI_U_0

Window A Higher 2 bits of Start Address for U plane

Makes the start address 34 bits by adding the higher 2 bits below.

Offset: 0x81b | Byte Offset: 0x206c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	A_START_ADDR_FIELD2_HI_U: Window A Start Address for U plane. This is a byte address.

21.10.29 DC_WINBUF_A_START_ADDR_FIELD2_HI_U_NS_0

Window A Shadowed Higher 32 bits of Start Address for U plane

Makes the start address 34 bits by adding the higher 2 bits below.

Offset: 0x81c | Byte Offset: 0x2070 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	A_START_ADDR_FIELD2_HI_U_NS: Window A Shadowed Higher 2 bits of Start Address for U plane.

Bit	Reset	Description
		This is the ARM set shadow register of the U start address.

21.10.30 DC_WINBUF_A_START_ADDR_FIELD2_HI_V_0

Window A Higher 2 bits of Start Address for V plane

Makes the start address 34 bits by adding the higher 2 bits below.

Offset: 0x81d | Byte Offset: 0x2074 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	A_START_ADDR_FIELD2_HI_V: Window A Higher 2 bits of Start Address for V plane. This is a byte address.

21.10.31 DC_WINBUF_A_START_ADDR_FIELD2_HI_V_NS_0

Window A Shadowed Higher 2 bits of Start Address for V plane

Makes the start address 34 bits by adding the higher 2 bits below.

Offset: 0x81e | Byte Offset: 0x2078 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	A_START_ADDR_FIELD2_HI_V_NS: Window A Shadowed Higher 2 bits of Start Address for V plane. This is the ARM set shadow register of the U start address.

21.10.32 DC_WINBUF_A_ADDR_H_OFFSET_FIELD2_0

Window A Horizontal address offset

Offset: 0x81f | Byte Offset: 0x207c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_ADDR_H_OFFSET_FIELD2: Window A Horizontal address offset. This is a byte address. The LSB is not used for the 16-bpp non-planar pixel format, and the last 2 LSBs are not used for the 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies the horizontal offset of the Y plane. The horizontal offsets of U/V plane are derived by hardware.

21.10.33 DC_WINBUF_A_ADDR_H_OFFSET_FIELD2_NS_0

Window A Shadowed Horizontal address offset

Offset: 0x820 | Byte Offset: 0x2080 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_ADDR_H_OFFSET_FIELD2_NS: Window A Shadowed Horizontal address offset. This is the ARM set shadow of ADDR_H_OFFSET.

21.10.34 DC_WINBUF_A_ADDR_V_OFFSET_FIELD2_0

Window A Vertical address offset

Offset: 0x821 | Byte Offset: 0x2084 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_ADDR_V_OFFSET_FIELD2: Window A Vertical address offset. This is the vertical coordinate. For YUV planar pixel format, this specifies the vertical coordinate of the Y plane. The vertical coordinate of U/V plane is derived by hardware.

21.10.35 DC_WINBUF_A_ADDR_V_OFFSET_FIELD2_NS_0

Window A Shadowed Vertical address offset

Offset: 0x822 | Byte Offset: 0x2088 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_ADDR_V_OFFSET_FIELD2_NS: Window A Shadowed Vertical address offset. This is the ARM set shadow of ADDR_V_OFFSET

21.10.36 DC_WINBUF_A_UFLOW_CTRL_0

When the underflow debug mode is enabled. Instead of sending out the last valid pixel, DBG_PIXEL is sent out.

Offset: 0x824 | Byte Offset: 0x2090 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	A_UFLOW_CTRL_DBG_MODE: 0 = DISABLE 1 = ENABLE

21.10.37 DC_WINBUF_A_UFLOW_DBG_PIXEL_0

Offset: 0x825 | Byte Offset: 0x2094 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_UFLOW_DBG_PIXEL

21.10.38 DC_WINBUF_A_UFLOW_THRESHOLD_0

Offset: 0x826 | Byte Offset: 0x2098 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxx0000000000000)

Bit	Reset	Description
12:0	0x0	A_UFLOW_THRESHOLD

21.10.39 DC_WINBUF_A_SPOOL_UP_0

Spool up time configuration for different windows related logic.

SPOOL_UP_CTRL = MAX - This is the current Tegra K1 behavior. This allows for memfetch to start fetching for a window (irrespective of its relative position on the screen), at the very beginning of the Vblank of the next frame. Since the spool up time is maximum, this should have the least probability of underflow of the display pipe. However, on the down side, since all three windows would start fetching at the same time, and since display is a high priority client for MC, it has a high potential to starve out other clients.

SPOOL_UP_CTRL = PROGRAMMABLE, SPOOL_UP_DURATION = 1 - This corresponds to the Tegra 3 behavior. Since the fetch starts only 1 line prior to the active scan line of a window, this has high probability of causing underflow, but has a lesser impact on the other MC clients.

So, this register programmability programs a method to modulate the spool up time between the best and worst, if required, when other MC clients are present. For example when GPU/VIC is active, we may not want a very aggressive spool up and program some definite spool up duration. So, spool up time is a general knob, which would be a function of the resolution, bpp, active clients, memory speed, etc.

SPOOL_UP_EDGE :- By default the memfetch start signal generated by the hvtgen logic is one line clock wide.

For SPOOL_UP_CTRL = MAX:

- In this case if SPOOL_UP_EDGE is programmed to NEGEDGE. This ensures that if any act_req is sent during the time vcounter = 0(at the beginning of the frame). The activate value is loaded into the active state and the memfetch would see the latest updated value. The downside of programming it to NEGDGE is that we lose out on 1 line clock worth of spool up. This is required based upon Tegra compatibility and the way tests are written.
- In this case if SPOOL_UP_EDGE is programmed to POSEDGE. Then the memfetch would start fetching at the very beginning of the frame start pulse (1 line clock wide) and would give maximum spool up time. However, this may give test failure and potential Tegra compatibility issues as described above

FOR SPOOL_UP_CTRL = PROGRAMMABLE :

- Special care must be taken here. If SPOOL_UP_DURATION is set to 1. SPOOL_UP_EDGE must be set to POSEDGE to allow for at least 1 line worth of spool up.
- If SPOOL_UP_DURATION is > 1. Then SPOOL_UP_EDGE can be either POSEDGE or NEGEDGE. If it is NEGEDGE, then the effective spool up duration would be 1 line clock less than the SPOOL_UP_DURATION.

Offset: 0x827 │ Byte Offset: 0x209c │ Read/Write: R/W │Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
28:16	X	A_SPOOL_UP_DURATION
1	0x0	A_SPOOL_UP_EDGE: 0 = NEGEDGE 1 = POSEDGE
0	0x0	A_SPOOL_UP_CTRL: 0 = MAX 1 = PROGRAMMABLE

21.10.40 DC_WINBUF_A_SCALEFACTOR_THRESHOLD_0

Registers for latency allowance scaling. Threshold is in terms of 64 byte atoms. Display memfetch compares its buffer occupancy with the threshold and generates indication to MC if its buffer occupancy is above HWM or below LWM.

Offset: 0x828 │ Byte Offset: 0x20a0 │ Read/Write: R/W │ Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:16	0xffff	A_SF_LWM_THRESHOLD
15:0	0xffff	A_SF_HWM_THRESHOLD

21.10.41 DC_WINBUF_A_LATENCY_THRESHOLD_0

The LATENCY_THRESHOLD register controls the behavior of the rdy4_latency_event from the display to the MC.

If B_RDY4LATENCY_THRESHOLD_ENABLE is set to DISABLE, the sideband rdy4_latency_event is always asserted, indicating that the display is always ready for a DVFS event.

The B_RDY4LATENCY_THRESHOLD field indicates the required occupancy of line buffers of windows below which the sideband rdy4_latency_event is deasserted. The occupancy figure is in units of 64B atoms for memfetch windows and in units of scan lines for regular windows.

The B_RDY4LATENCY_SPOOLUP_DURATION field indicates the number of scan lines during when the sideband will remain asserted (DVFS is allowed). This takes affect only when B_RDY4LATENCY_SPOOLUP_CTRL is set to ALLOW.

If the B_RDY4LATENCY_SPOOLUP_CTRL field is set to DISALLOW, the above register field is not affected and DVFS is disallowed until the threshold is reached.

Offset: 0x829 │ Byte Offset: 0x20a4 │ Read/Write: R/W │ Reset: 0x0000ffff (0b00x00000000000001111111111111111)

Bit	Reset	Description
31	DISABLE	A_RDY4LATENCY_THRESHOLD_ENABLE: 0 = DISABLE 1 = ENABLE
30	DISALLOW	A_RDY4LATENCY_SPOOLUP_CTRL: 0 = DISALLOW 1 = ALLOW
28:16	0x0	A_RDY4LATENCY_SPOOLUP_DURATION
15:0	0xffff	A_RDY4LATENCY_THRESHOLD

21.10.42 DC_WINBUF_A_MEMFETCH_DEBUG_STATUS_0

Memfetch Debug Status

Offset: 0x82a │ Byte Offset: 0x20a8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000000000000)

Bit	Reset	Description
16	0x0	A_DEBUG_ENABLE:reg to enable FGCG for flops added to debug signals in memfetch.
15	0x0	B_ALIGN_FIFO_NON_IDLE: align FIFO idle at end of frame. 0= align FIFO is idle at end of frame. 1= align FIFO is not idle at end of frame
14	0x0	B_PIPE1_FIFO_NON_IDLE: pipe1 FIFO idle at end of frame. 0= pipe1 FIFO is idle at end of frame. 1= pipe1 FIFO is not idle at end of frame
13	0x0	B_PIPE0_FIFO_NON_IDLE: pipe0 FIFO idle at end of frame. 0= pipe0 FIFO is idle at end of frame. 1= pipe0 FIFO is not idle at end of frame
12	0x0	B_FRAME_FIFO_NON_IDLE: frame FIFO idle at end of frame. 0= frame FIFO is idle at end of frame. 1= frame FIFO is not idle at end of frame
11	0x0	B_SURFACE_FIFO_NON_IDLE: surface FIFO idle at end of frame. 0= surface FIFO is idle at end of frame. 1= surface FIFO is not idle at end of frame
10	0x0	B_TAG_FIFO_NON_IDLE: tag FIFO idle at end of frame. 0= tag FIFO is idle at end of frame. 1= tag FIFO is not idle at end of frame
9	0x0	B_RSP_FIFO_NON_IDLE: Response FIFO idle at end of frame. 0= response FIFO is idle at end of frame. 1= response FIFO is not idle at end of frame

Bit	Reset	Description
8	0x0	B_REQ_FIFO_NON_IDLE: Request FIFO idle at end of frame. 0= request FIFO is idle at end of frame. 1= request FIFO is not idle at end of frame
7	0x0	B_DP_VALID_NON_IDLE: DP lines valid at end of frame. 0= DP lines are not valid at end of frame. 1= DP lines are valid at end of frame
6	0x0	B_V_SSM_NON_IDLE: V SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame
5	0x0	B_U_SSM_NON_IDLE: U SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame
4	0x0	B_Y_SSM_NON_IDLE: Y SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame
3	0x0	B_DP_POOL_AVAIL: DP thinks there is data in the pool at end of frame. 0= No pool data available at end of frame. 1= pool data available at end of frame
2	0x0	B_POOL_NOT_EMPTY: Status of pool at end of frame. 0= pool is empty at end of frame. 1= pool is not empty at end of frame
1	0x0	B_UNDERFLOW_LINE1: Underflow of line0 0= No underflow 1= line0 underflowed
0	0x0	B_UNDERFLOW_LINE0: Underflow of line0 0= No underflow 1= line0 underflowed

21.10.43 DC_WINBUF_A_MEMFETCH_CONTROL_0

Note: MEMFETCH_CLK_GATE_OVERRIDE should only be enabled AFTER the vertical scaler settings in the active copy - VDDA_INCREMENT is properly set up.

Memfetch Control Register

Offset: 0x82b │ Byte Offset: 0x20ac │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00) | Default: 0x00000001

Bit	Reset	SW Default	Description
1	0x0	NONE	A_MEMFETCH_CLK_GATE_OVERRIDE: Disable clock-gating of memfetch for a controller (all three windows) 0 = DISABLE 1 = ENABLE
0	0x0	ENABLE	A_MEMFETCH_RESET: Reset memfetch for a controller (all three windows) 0 = DISABLE 1 = ENABLE

21.10.44 DC_WINBUF_A_OCCUPANCY_THROTTLE_0

Offset: 0x82c | Byte Offset: 0x20b0 | Read/Write: R/W | Reset: 0xffff0000 (0b1111111111111111xxxxxxxxxxxxxxx0)

Bit	Reset	Description
31:16	0xffff	A_OCCUPANCY_MAX_THRESHOLD
0	0x0	A_OCCUPANCY_THROTTLE_MODE: 0 = DISABLE 1 = ENABLE

21.10.45 DC_WINBUF_A_SCRATCH_REGISTER_0_0

Offset: 0x82d | Byte Offset: 0x20b4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	A_SCRATCH_REGISTER_0:Scratch register 0

21.10.46 DC_WINBUF_A_SCRATCH_REGISTER_1_0

Offset: 0x82e | Byte Offset: 0x20b8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	A_SCRATCH_REGISTER_1:Scratch register 1

21.11 Window B (WINC_B) Registers

These registers control window B parameters.

21.11.1 DC_WINC_B_COLOR_PALETTE_0

Window B Color Palette

This is used for palletized data format (color depth of 8-bpp or less) or for gamma correction for non-palletized data formats (color depth of more than 8-bpp).

Each window has its own color palette which consists of three 256x8 register file which can be written by host and indexed (read) by the window.

For palletized data format less than 8-bpp, the pixel data is aligned to least significant bits of the palette index (address) and the remaining upper bits are filled with the corresponding bits of the Palette Color Extension. For example, for 4-bpp mode, the pixel data occupies bits 3-0 of the palette index, and bits 7-4 of the palette index are set to bits 7-4 of the Palette Color Extension.

Note that host read is assumed to be not needed - software can cache the color palette in system memory.

This is an array of 256 identical register entries; the register fields below apply to each entry.

Offset: 0x500..0x5ff | Byte Offset: 0x1400..0x17fc | Read/Write: WO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	B_COLOR_PALETTE_B: Blue Color Palette
15:8	X	B_COLOR_PALETTE_G: Green Color Palette

Bit	Reset	Description
7:0	X	B_COLOR_PALETTE_R: Red Color Palette

21.11.2 DC_WINC_B_PALETTE_COLOR_EXT_0

Window B Palette Color Extension

Palette extension for 1-bpp, 2-bpp, and 4-bpp. These bits provide the upper most significant bits for indexing the color palette.

Offset: 0x600 │ Byte Offset: 0x1800 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:1	X	B_PALETTE_COLOR_EXT: Window B Palette Color Extension bits 7-1 are used for 1-bpp mode bits 7-2 are used for 2-bpp mode bits 7-4 are used for 4-bpp mode

21.11.3 DC_WINC_B_H_FILTER_P00_0

Window B Horizontal Filter phase 00

Horizontal Scaling Filter Coefficients

Horizontal scaling filter is a 6-tap filter with 4-bit positional phase.

- Coefficients 0 and 5 are 3-bit signed values ranging from -4 to 3.
- Coefficients 1 and 4 are 5-bit signed values ranging from -16 to 15.
- Coefficients 2 and 3 are 8-bit unsigned values ranging from 0 to 128.
- Coefficient 0 is the multiplier for the earliest pixel (P0) in the group of 6 pixels and coefficient 5 is the multiplier for the latest pixel (P5) in the group. The output pixel positional phase is defined as centered in P2 if the positional phase is 0 or proportionally in between P2 and P3 if the positional phase is larger than 0.

Sum of all coefficients for each phase should be 128 typically, and software should never program the sum of all coefficients for a phase to be more than 128. For each horizontal positional phase, the 6 filter coefficients requires 32 register bits. Note that color value ranges from 0 to 255 for Y, R, G, B, and -128 to 127 for U and V.

Offset: 0x601 │ Byte Offset: 0x1804 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P00C5: Phase 00 coefficient 5 (typically 0)
28:24	X	B_H_FILTER_P00C4: Phase 00 coefficient 4 (typically 0)
23:16	X	B_H_FILTER_P00C3: Phase 00 coefficient 3 (typically 0)
15:8	X	B_H_FILTER_P00C2: Phase 00 coefficient 2 (typically 128)
7:3	X	B_H_FILTER_P00C1: Phase 00 coefficient 1 (typically 0)
2:0	X	B_H_FILTER_P00C0: Phase 00 coefficient 0 (typically 0)

21.11.4 DC_WINC_B_H_FILTER_P01_0

Window B Horizontal Filter phase 01

Offset: 0x602 | Byte Offset: 0x1808 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P01C5: Phase 01 coefficient 5 (typically 1)
28:24	X	B_H_FILTER_P01C4: Phase 01 coefficient 4 (typically -2)
23:16	X	B_H_FILTER_P01C3: Phase 01 coefficient 3 (typically 8)
15:8	X	B_H_FILTER_P01C2: Phase 01 coefficient 2 (typically 124)
7:3	X	B_H_FILTER_P01C1: Phase 01 coefficient 1 (typically -4)
2:0	X	B_H_FILTER_P01C0: Phase 01 coefficient 0 (typically 1)

21.11.5 DC_WINC_B_H_FILTER_P02_0

Window B Horizontal Filter phase 02

Offset: 0x603 | Byte Offset: 0x180c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P02C5: Phase 02 coefficient 5 (typically 1)
28:24	X	B_H_FILTER_P02C4: Phase 02 coefficient 4 (typically -5)
23:16	X	B_H_FILTER_P02C3: Phase 02 coefficient 3 (typically 17)
15:8	X	B_H_FILTER_P02C2: Phase 02 coefficient 2 (typically 122)
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)

21.11.6 DC_WINC_B_H_FILTER_P03_0

Window B Horizontal Filter phase 03

Offset: 0x604 | Byte Offset: 0x1810 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P03C5: Phase 03 coefficient 5 (typically 2)
28:24	X	B_H_FILTER_P03C4: Phase 03 coefficient 4 (typically -7)
23:16	X	B_H_FILTER_P03C3: Phase 03 coefficient 3 (typically 27)
15:8	X	B_H_FILTER_P03C2: Phase 03 coefficient 2 (typically 115)
7:3	X	B_H_FILTER_P03C1: Phase 03 coefficient 1 (typically -11)
2:0	X	B_H_FILTER_P03C0: Phase 03 coefficient 0 (typically 2)

21.11.7 DC_WINC_B_H_FILTER_P04_0

Window B Horizontal Filter phase 04

Offset: 0x605 | Byte Offset: 0x1814 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P04C5: Phase 04 coefficient 5 (typically 2)
28:24	X	B_H_FILTER_P04C4: Phase 04 coefficient 4 (typically -9)
23:16	X	B_H_FILTER_P04C3: Phase 04 coefficient 3 (typically 37)
15:8	X	B_H_FILTER_P04C2: Phase 04 coefficient 2 (typically 109)
7:3	X	B_H_FILTER_P04C1: Phase 04 coefficient 1 (typically -13)
2:0	X	B_H_FILTER_P04C0: Phase 04 coefficient 0 (typically 2)

21.11.8 DC_WINC_B_H_FILTER_P05_0

Window B Horizontal Filter phase 05

Offset: 0x606 | Byte Offset: 0x1818 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P05C5: Phase 05 coefficient 5 (typically 2)
28:24	X	B_H_FILTER_P05C4: Phase 05 coefficient 4 (typically -11)
23:16	X	B_H_FILTER_P05C3: Phase 05 coefficient 3 (typically 47)
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)

21.11.9 DC_WINC_B_H_FILTER_P06_0

Window B Horizontal Filter phase 06

Offset: 0x607 | Byte Offset: 0x181c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P06C5: Phase 06 coefficient 5 (typically 3)
28:24	X	B_H_FILTER_P06C4: Phase 06 coefficient 4 (typically -13)
23:16	X	B_H_FILTER_P06C3: Phase 06 coefficient 3 (typically 56)
15:8	X	B_H_FILTER_P06C2: Phase 06 coefficient 2 (typically 94)
7:3	X	B_H_FILTER_P06C1: Phase 06 coefficient 1 (typically -15)
2:0	X	B_H_FILTER_P06C0: Phase 06 coefficient 0 (typically 3)

21.11.10 DC_WINC_B_H_FILTER_P07_0

Window B Horizontal Filter phase 07

Offset: 0x608 | Byte Offset: 0x1820 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P07C5: Phase 07 coefficient 5 (typically 3)
28:24	X	B_H_FILTER_P07C4: Phase 07 coefficient 4 (typically -14)
23:16	X	B_H_FILTER_P07C3: Phase 07 coefficient 3 (typically 67)
15:8	X	B_H_FILTER_P07C2: Phase 07 coefficient 2 (typically 85)
7:3	X	B_H_FILTER_P07C1: Phase 07 coefficient 1 (typically -16)
2:0	X	B_H_FILTER_P07C0: Phase 07 coefficient 0 (typically 3)

21.11.11 DC_WINC_B_H_FILTER_P08_0

Window B Horizontal Filter phase 08

Offset: 0x609 | Byte Offset: 0x1824 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P08C5: Phase 08 coefficient 5 (typically 3)
28:24	X	B_H_FILTER_P08C4: Phase 08 coefficient 4 (typically -15)
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)

21.11.12 DC_WINC_B_H_FILTER_P09_0

Window B Horizontal Filter phase 09

Offset: 0x60a | Byte Offset: 0x1828 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P09C5: Phase 09 coefficient 5 (typically 3)
28:24	X	B_H_FILTER_P09C4: Phase 09 coefficient 4 (typically -16)
23:16	X	B_H_FILTER_P09C3: Phase 09 coefficient 3 (typically 85)
15:8	X	B_H_FILTER_P09C2: Phase 09 coefficient 2 (typically 67)
7:3	X	B_H_FILTER_P09C1: Phase 09 coefficient 1 (typically -14)
2:0	X	B_H_FILTER_P09C0: Phase 09 coefficient 0 (typically 3)

21.11.13 DC_WINC_B_H_FILTER_P0A_0

Window B Horizontal Filter phase 0A

Offset: 0x60b │ Byte Offset: 0x182c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P0AC5: Phase 0A coefficient 5 (typically 3)
28:24	X	B_H_FILTER_P0AC4: Phase 0A coefficient 4 (typically -15)
23:16	X	B_H_FILTER_P0AC3: Phase 0A coefficient 3 (typically 94)
15:8	X	B_H_FILTER_P0AC2: Phase 0A coefficient 2 (typically 56)
7:3	X	B_H_FILTER_P0AC1: Phase 0A coefficient 1 (typically -13)
2:0	X	B_H_FILTER_P0AC0: Phase 0A coefficient 0 (typically 3)

21.11.14 DC_WINC_B_H_FILTER_P0B_0

Window B Horizontal Filter phase 0B

Offset: 0x60c │ Byte Offset: 0x1830 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P0BC5: Phase 0B coefficient 5 (typically 3)
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)

21.11.15 DC_WINC_B_H_FILTER_P0C_0

Window B Horizontal Filter phase 0C

Offset: 0x60d │ Byte Offset: 0x1834 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P0CC5: Phase 0C coefficient 5 (typically 2)
28:24	X	B_H_FILTER_P0CC4: Phase 0C coefficient 4 (typically -13)
23:16	X	B_H_FILTER_P0CC3: Phase 0C coefficient 3 (typically 109)
15:8	X	B_H_FILTER_P0CC2: Phase 0C coefficient 2 (typically 37)
7:3	X	B_H_FILTER_P0CC1: Phase 0C coefficient 1 (typically -9)
2:0	X	B_H_FILTER_P0CC0: Phase 0C coefficient 0 (typically 2)

21.11.16 DC_WINC_B_H_FILTER_P0D_0

Window B Horizontal Filter phase 0D

Offset: 0x60e │ Byte Offset: 0x1838 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P0DC5: Phase 0D coefficient 5 (typically 2)
28:24	X	B_H_FILTER_P0DC4: Phase 0D coefficient 4 (typically -11)
23:16	X	B_H_FILTER_P0DC3: Phase 0D coefficient 3 (typically 115)
15:8	X	B_H_FILTER_P0DC2: Phase 0D coefficient 2 (typically 27)
7:3	X	B_H_FILTER_P0DC1: Phase 0D coefficient 1 (typically -7)
2:0	X	B_H_FILTER_P0DC0: Phase 0D coefficient 0 (typically 2)

21.11.17 DC_WINC_B_H_FILTER_P0E_0

Window B Horizontal Filter phase 0E

Offset: 0x60f │ Byte Offset: 0x183c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P0EC5: Phase 0E coefficient 5 (typically 1)
28:24	X	B_H_FILTER_P0EC4: Phase 0E coefficient 4 (typically -8)
23:16	X	B_H_FILTER_P0EC3: Phase 0E coefficient 3 (typically 122)
15:8	X	B_H_FILTER_P0EC2: Phase 0E coefficient 2 (typically 17)
7:3	X	B_H_FILTER_P0EC1: Phase 0E coefficient 1 (typically -5)
2:0	X	B_H_FILTER_P0EC0: Phase 0E coefficient 0 (typically 1)

21.11.18 DC_WINC_B_H_FILTER_P0F_0

Window B Horizontal Filter phase 0F

Offset: 0x610 │ Byte Offset: 0x1840 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P0FC5: Phase 0F coefficient 5 (typically 1)
28:24	X	B_H_FILTER_P0FC4: Phase 0F coefficient 4 (typically -4)
23:16	X	B_H_FILTER_P0FC3: Phase 0F coefficient 3 (typically 124)
15:8	X	B_H_FILTER_P0FC2: Phase 0F coefficient 2 (typically 8)
7:3	X	B_H_FILTER_P0FC1: Phase 0F coefficient 1 (typically -2)
2:0	X	B_H_FILTER_P0FC0: Phase 0F coefficient 0 (typically 1)

21.11.19 DC_WINC_B_CSC_YOF_0

Window B CSC Y Offset

Color Space Conversion coefficients.

The CSC can be used for YUV to RGB conversion with brightness and hue/saturation control.

The CSC can only be enabled for window B controlled by CSC_ENABLE register bits.

For Y color, the Y offset is applied first and saturation (clipping) is performed immediately after the Y offset is applied.

R = sat(KYRGB * sat(Y + YOF) + KUR * U + KVR * V)

G = sat(KYRGB * sat(Y + YOF) + KUG * U + KVG * V)

B = sat(KYRGB * sat(Y + YOF) + KUB * U + KVB * V)

Saturation and rounding is performed in the range of 0 to 255 for the above equations.

Typical values are:

YOF = -16.000, KYRGB = 1.1644

KUR = 0.0000, KVR = 1.5960

KUG = -0.3918, KVG = -0.8130

KUB = 2.0172, KVB = 0.0000

KUR and KVB are typically 0.0000 but they may be programmed non-zero for hue rotation.

The CSC can also take RGB input, in which case YOF, KVB, KUG, KUR should be programmed to 0, and KYRGB will be forced to 0 by the hardware for generating R and B. KYRGB will not be forced to 0 for generating G. KVR, KYRGB, and KUB can be programmed to 1.0 or used as gain control for R, G, B, correspondingly.

Note that color value ranges from 0 to 255 for Y, R, G, B and -128 to 127 for U and V.

Offset: 0x611 │ Byte Offset: 0x1844 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_CSC_YOF: Y Offset in s.7.0 format

21.11.20 DC_WINC_B_CSC_KYRGB_0

Window B CSC Y Coefficient (gain) for RGB

Offset: 0x612 │ Byte Offset: 0x1848 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	B_CSC_KYRGB: Y Gain for R, G, B colors in 2.8 format

21.11.21 DC_WINC_B_CSC_KUR_0

Window B CSC U coefficient for R

Offset: 0x613 | Byte Offset: 0x184c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	B_CSC_KUR: U coefficients for R in s.2.8 format

21.11.22 DC_WINC_B_CSC_KVR_0

Window B CSC V coefficient for R

Offset: 0x614 | Byte Offset: 0x1850 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	B_CSC_KVR: V coefficients for R in s.2.8 format

21.11.23 DC_WINC_B_CSC_KUG_0

Window B CSC U coefficient for G

Offset: 0x615 | Byte Offset: 0x1854 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	B_CSC_KUG: U coefficients for G in s.1.8 format

21.11.24 DC_WINC_B_CSC_KVG_0

Window B CSC V coefficient for G

Offset: 0x616 | Byte Offset: 0x1858 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	B_CSC_KVG: V coefficients for G in s.1.8 format

21.11.25 DC_WINC_B_CSC_KUB_0

Window B CSC U coefficient for B

Offset: 0x617 | Byte Offset: 0x185c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	B_CSC_KUB: U coefficients for B in s.2.8 format

21.11.26 DC_WINC_B_CSC_KVB_0

Window B CSC V coefficient for B

Offset: 0x618 | Byte Offset: 0x1860 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	B_CSC_KVB: V coefficients for B in s.2.8 format

21.11.27 DC_WINC_B_V_FILTER_P00_0

Window B Vertical Filter phase 00

Vertical scaling filter coefficients

Vertical scaling filter is a 2-tap filter with 4-bit positional phase.

Coefficients 0 and 1 are 8-bit unsigned values ranging from 0 to 128.

Coefficient 0 is the multiplier for the earlier pixel (P0) in the group of 2 pixels and coefficient 1 is the multiplier for the later pixel (P1) in the group. The output pixel positional phase is defined as centered in P0 if the positional phase is 0 or proportionally in between P0 and P1 if the positional phase is larger than 0.

Sum of all coefficients for each phase should be 128 typically therefore coefficient 1 can be calculated from (1 - coefficient 0) and therefore only coefficient 0 is programmed.

For each vertical positional phase, the filter coefficient requires 8 register bits. Note that color value ranges from 0 to 255 for Y, R, G, B, and -128 to 127 for U and V.

Offset: 0x619 | Byte Offset: 0x1864 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P00C0: Phase 00 coefficient 0 (typically 128)

21.11.28 DC_WINC_B_V_FILTER_P01_0

Window B Vertical Filter phase 01

Offset: 0x61a | Byte Offset: 0x1868 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P01C0: Phase 01 coefficient 0 (typically 120)

21.11.29 DC_WINC_B_V_FILTER_P02_0

Window B Vertical Filter phase 02

Offset: 0x61b | Byte Offset: 0x186c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P02C0: Phase 02 coefficient 0 (typically 112)

21.11.30 DC_WINC_B_V_FILTER_P03_0

Window B Vertical Filter phase 03

Offset: 0x61c | Byte Offset: 0x1870 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P03C0: Phase 03 coefficient 0 (typically 104)

21.11.31 DC_WINC_B_V_FILTER_P04_0

Window B Vertical Filter phase 04

Offset: 0x61d | Byte Offset: 0x1874 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P04C0: Phase 04 coefficient 0 (typically 96)

21.11.32 DC_WINC_B_V_FILTER_P05_0

Window B Vertical Filter phase 05

Offset: 0x61e | Byte Offset: 0x1878 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P05C0: Phase 05 coefficient 0 (typically 88)

21.11.33 DC_WINC_B_V_FILTER_P06_0

Window B Vertical Filter phase 06

Offset: 0x61f | Byte Offset: 0x187c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P06C0: Phase 06 coefficient 0 (typically 80)

21.11.34 DC_WINC_B_V_FILTER_P07_0

Window B Vertical Filter phase 07

Offset: 0x620 | Byte Offset: 0x1880 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P07C0: Phase 07 coefficient 0 (typically 72)

21.11.35 DC_WINC_B_V_FILTER_P08_0

Window B Vertical Filter phase 08

Offset: 0x621 | Byte Offset: 0x1884 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P08C0: Phase 08 coefficient 0 (typically 64)

21.11.36 DC_WINC_B_V_FILTER_P09_0

Window B Vertical Filter phase 09

Offset: 0x622 | Byte Offset: 0x1888 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P09C0: Phase 09 coefficient 0 (typically 56)

21.11.37 DC_WINC_B_V_FILTER_P0A_0

Window B Vertical Filter phase 0A

Offset: 0x623 | Byte Offset: 0x188c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P0AC0: Phase 0A coefficient 0 (typically 48)

21.11.38 DC_WINC_B_V_FILTER_P0B_0

Window B Vertical Filter phase 0B

Offset: 0x624 | Byte Offset: 0x1890 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P0BC0: Phase 0B coefficient 0 (typically 40)

21.11.39 DC_WINC_B_V_FILTER_P0C_0

Window B Vertical Filter phase 0C

Offset: 0x625 | Byte Offset: 0x1894 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P0CC0: Phase 0C coefficient 0 (typically 32)

21.11.40 DC_WINC_B_V_FILTER_P0D_0

Window B Vertical Filter phase 0D

Offset: 0x626 | Byte Offset: 0x1898 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P0DC0: Phase 0D coefficient 0 (typically 24)

21.11.41 DC_WINC_B_V_FILTER_P0E_0

Window B Vertical Filter phase 0E

Offset: 0x627 | Byte Offset: 0x189c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P0EC0: Phase 0E coefficient 0 (typically 16)

21.11.42 DC_WINC_B_V_FILTER_P0F_0

Window B Vertical Filter phase 0F

Offset: 0x628 | Byte Offset: 0x18a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P0FC0: Phase 0F coefficient 0 (typically 8)

21.11.43 DC_WINC_B_H_FILTER_HI_P00_0

Window B Horizontal Filter phase 00

Offset: 0x629 | Byte Offset: 0x18a4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	B_H_FILTER_HI_P00C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	B_H_FILTER_HI_P00C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	B_H_FILTER_HI_P00C3: MSB 9:9 of the lobe 3.
4	X	B_H_FILTER_HI_P00C2: MSB 9:9 of the lobe 2.
3:2	X	B_H_FILTER_HI_P00C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	B_H_FILTER_HI_P00C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.11.44 DC_WINC_B_H_FILTER_HI_P01_0

Window B Horizontal Filter phase 01

Offset: 0x62a | Byte Offset: 0x18a8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	B_H_FILTER_HI_P01C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	B_H_FILTER_HI_P01C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	B_H_FILTER_HI_P01C3: MSB 9:9 of the lobe 3.
4	X	B_H_FILTER_HI_P01C2: MSB 9:9 of the lobe 2.
3:2	X	B_H_FILTER_HI_P01C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	B_H_FILTER_HI_P01C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.11.45 DC_WINC_B_H_FILTER_HI_P02_0

Window B Horizontal Filter phase 02

Offset: 0x62b | Byte Offset: 0x18ac | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	B_H_FILTER_HI_P02C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	B_H_FILTER_HI_P02C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	B_H_FILTER_HI_P02C3: MSB 9:9 of the lobe 3.
4	X	B_H_FILTER_HI_P02C2: MSB 9:9 of the lobe 2.
3:2	X	B_H_FILTER_HI_P02C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	B_H_FILTER_HI_P02C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.11.46 DC_WINC_B_H_FILTER_HI_P03_0

Window B Horizontal Filter phase 03

Offset: 0x62c | Byte Offset: 0x18b0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	B_H_FILTER_HI_P03C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	B_H_FILTER_HI_P03C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	B_H_FILTER_HI_P03C3: MSB 9:9 of the lobe 3.
4	X	B_H_FILTER_HI_P03C2: MSB 9:9 of the lobe 2.
3:2	X	B_H_FILTER_HI_P03C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.

Bit	Reset	Description
1:0	X	B_H_FILTER_HI_P03C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.11.47 DC_WINC_B_H_FILTER_HI_P04_0

Window B Horizontal Filter phase 04

Offset: 0x62d | Byte Offset: 0x18b4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	B_H_FILTER_HI_P04C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	B_H_FILTER_HI_P04C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	B_H_FILTER_HI_P04C3: MSB 9:9 of the lobe 3.
4	X	B_H_FILTER_HI_P04C2: MSB 9:9 of the lobe 2.
3:2	X	B_H_FILTER_HI_P04C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	B_H_FILTER_HI_P04C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.11.48 DC_WINC_B_H_FILTER_HI_P05_0

Window B Horizontal Filter phase 05

Offset: 0x62e | Byte Offset: 0x18b8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	B_H_FILTER_HI_P05C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	B_H_FILTER_HI_P05C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	B_H_FILTER_HI_P05C3: MSB 9:9 of the lobe 3.
4	X	B_H_FILTER_HI_P05C2: MSB 9:9 of the lobe 2.
3:2	X	B_H_FILTER_HI_P05C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	B_H_FILTER_HI_P05C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.11.49 DC_WINC_B_H_FILTER_HI_P06_0

Window B Horizontal Filter phase 06

Offset: 0x62f | Byte Offset: 0x18bc | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	B_H_FILTER_HI_P06C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in the B_H_FILTER_P** register.
7:6	X	B_H_FILTER_HI_P06C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.

Bit	Reset	Description
5	X	B_H_FILTER_HI_P06C3: MSB 9:9 of the lobe 3.
4	X	B_H_FILTER_HI_P06C2: MSB 9:9 of the lobe 2.
3:2	X	B_H_FILTER_HI_P06C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	B_H_FILTER_HI_P06C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in the B_H_FILTER_P** register.

21.11.50 DC_WINC_B_H_FILTER_HI_P07_0

Window B Horizontal Filter phase 07

Offset: 0x630 | Byte Offset: 0x18c0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	B_H_FILTER_HI_P07C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	B_H_FILTER_HI_P07C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	B_H_FILTER_HI_P07C3: MSB 9:9 of the lobe 3.
4	X	B_H_FILTER_HI_P07C2: MSB 9:9 of the lobe 2.
3:2	X	B_H_FILTER_HI_P07C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	B_H_FILTER_HI_P07C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.11.51 DC_WINC_B_H_FILTER_HI_P08_0

Window B Horizontal Filter phase 08

Offset: 0x631 | Byte Offset: 0x18c4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	B_H_FILTER_HI_P08C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	B_H_FILTER_HI_P08C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	B_H_FILTER_HI_P08C3: MSB 9:9 of the lobe 3.
4	X	B_H_FILTER_HI_P08C2: MSB 9:9 of the lobe 2.
3:2	X	B_H_FILTER_HI_P08C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	B_H_FILTER_HI_P08C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.11.52 DC_WINC_B_H_FILTER_HI_P09_0

Window B Horizontal Filter phase 09

Offset: 0x632 | Byte Offset: 0x18c8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	B_H_FILTER_HI_P09C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	B_H_FILTER_HI_P09C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	B_H_FILTER_HI_P09C3: MSB 9:9 of the lobe 3.
4	X	B_H_FILTER_HI_P09C2: MSB 9:9 of the lobe 2.
3:2	X	B_H_FILTER_HI_P09C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	B_H_FILTER_HI_P09C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.11.53 DC_WINC_B_H_FILTER_HI_P0A_0

Window B Horizontal Filter phase 0A

Offset: 0x633 | Byte Offset: 0x18cc | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	B_H_FILTER_HI_P0AC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	B_H_FILTER_HI_P0AC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	B_H_FILTER_HI_P0AC3: MSB 9:9 of the lobe 3.
4	X	B_H_FILTER_HI_P0AC2: MSB 9:9 of the lobe 2.
3:2	X	B_H_FILTER_HI_P0AC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	B_H_FILTER_HI_P0AC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.11.54 DC_WINC_B_H_FILTER_HI_P0B_0

Window B Horizontal Filter phase 0B

Offset: 0x634 | Byte Offset: 0x18d0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	B_H_FILTER_HI_P0BC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	B_H_FILTER_HI_P0BC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	B_H_FILTER_HI_P0BC3: MSB 9:9 of the lobe 3.
4	X	B_H_FILTER_HI_P0BC2: MSB 9:9 of the lobe 2.
3:2	X	B_H_FILTER_HI_P0BC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.

Bit	Reset	Description
1:0	X	B_H_FILTER_HI_P0BC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.11.55 DC_WINC_B_H_FILTER_HI_P0C_0

Window B Horizontal Filter phase 0C

Offset: 0x635 | Byte Offset: 0x18d4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	B_H_FILTER_HI_P0CC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in the B_H_FILTER_P** register.
7:6	X	B_H_FILTER_HI_P0CC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	B_H_FILTER_HI_P0CC3: MSB 9:9 of the lobe 3.
4	X	B_H_FILTER_HI_P0CC2: MSB 9:9 of the lobe 2.
3:2	X	B_H_FILTER_HI_P0CC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	B_H_FILTER_HI_P0CC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in the B_H_FILTER_P** register.

21.11.56 DC_WINC_B_H_FILTER_HI_P0D_0

Window B Horizontal Filter phase 0D

Offset: 0x636 | Byte Offset: 0x18d8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	B_H_FILTER_HI_P0DC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	B_H_FILTER_HI_P0DC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	B_H_FILTER_HI_P0DC3: MSB 9:9 of the lobe 3.
4	X	B_H_FILTER_HI_P0DC2: MSB 9:9 of the lobe 2.
3:2	X	B_H_FILTER_HI_P0DC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	B_H_FILTER_HI_P0DC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.11.57 DC_WINC_B_H_FILTER_HI_P0E_0

Window B Horizontal Filter phase 0E

Offset: 0x637 | Byte Offset: 0x18dc | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	B_H_FILTER_HI_P0EC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	B_H_FILTER_HI_P0EC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.

Bit	Reset	Description
5	X	B_H_FILTER_HI_P0EC3: MSB 9:9 of the lobe 3.
4	X	B_H_FILTER_HI_P0EC2: MSB 9:9 of the lobe 2.
3:2	X	B_H_FILTER_HI_P0EC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	B_H_FILTER_HI_P0EC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.11.58 DC_WINC_B_H_FILTER_HI_P0F_0

Window B Horizontal Filter phase 0F

Offset: 0x638 | Byte Offset: 0x18e0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	B_H_FILTER_HI_P0FC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	B_H_FILTER_HI_P0FC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	B_H_FILTER_HI_P0FC3: MSB 9:9 of the lobe 3.
4	X	B_H_FILTER_HI_P0FC2: MSB 9:9 of the lobe 2.
3:2	X	B_H_FILTER_HI_P0FC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	B_H_FILTER_HI_P0FC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

The registers under DC_WIN are double buffered

21.11.59 DC_WIN_B_WIN_OPTIONS_0

Window B Options

Class: Display Window Settings

Display Window B parameters

Offset: 0x700 | Byte Offset: 0x1c00 | Read/Write: R/W | Reset: 0x00XXXXXX (0b00xxxxxx0xxxxxx0x0xxxxxxxxx0xxxx)

Bit	Reset	Description
31	0x0	B_H_FILTER_MODE: Horizontal filter mode. 0 = With previous Tegra horizontal coeffients widths. 0,5 - signed 5 bits 1,4 - signed 9 bits 2,3 - unsigned 9 bits 1 = New mode. With the expanded horizontal filter coefficients widths. 0,5- signed 5 bits 1,4- signed 7 bits 2,3- unsigned 9 bits 0 = OLD 1 = NEW
30	0x0	B_WIN_ENABLE: Window B Window enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
23	0x0	B_INTERLACE_ENABLE: Window B Interlace enable. This controls the fetch unit toggle between odd and even (normal) base addresses at the start of each field. Base address select between odd/even fields is determined by HVTGEN field status. 0 = DISABLE 1 = ENABLE
22	X	B_YUV_RANGE_EXPAND: Window B Enable range expansion in the cases where RANGEREDFRM is 1 from mpd. Formula: Y = clip((Y-128)*2 + 128); Cb = clip((Cb-128)*2 + 128); Cr = clip((Cr-128)*2 + 128); where clip() function clips between 0 and 255. 0 = DISABLE 1 = ENABLE
20	X	B_DV_ENABLE: Window B Digital Vibrance Enable 0 = DISABLE 1 = ENABLE
18	X	B_CSC_ENABLE: Window B Color Space Conversion Enable. This controls the color space conversion and should be enabled for YCbCr/YUV color modes for conversion to B8G8R8 and for hue and saturation control. This can also be used for gain control for RGB color modes 0 = DISABLE 1 = ENABLE
16	0x0	B_CP_ENABLE: Window B Color Palette Enable. This controls the color palette and should be enabled for palletized color modes. For non-palletized color modes, the color palette can be enabled for gamma correction. 0 = DISABLE 1 = ENABLE
14	0x0	B_V_FILTER_UV_ALIGN: Window B V Filter UV Alignment. This is effective only when vertical scaling filter is enabled and only on these formats YCbCr420P, YUV420P, YCbCr422R, YUV422R, YCbCr422RA YUV422RA. When UV alignment is enabled, the chroma components are aligned to the even number of luma component lines. When disabled the chroma components are aligned to half a pixel below the corresponding even number of luma component lines. It is usually disabled unless the incoming video stream specifically indicates otherwise. 0 = DISABLE 1 = ENABLE
12	X	B_V_FILTER_OPTIMIZE: Window B V Filter Optimization. This is effective only when vertical scaling filter is enabled. This can be used (enabled) to temporarily disable the vertical scaling filter when the vertical scaling DDA fraction is zero. In this case the next line is not fetched from memory to save bandwidth and power. This feature cannot be used in 420P/422R/422RA formats. 0 = DISABLE 1 = ENABLE
10	X	B_V_FILTER_ENABLE: Window B V Filter Enable. This controls V scaling filter and is effective only for non-palletized color modes. If V filter is disabled, only one line is read from memory for each output line. NOTE: This feature is not supported. This bit is ignored and is a don't care. 0 = DISABLE 1 = ENABLE
8	X	B_H_FILTER_ENABLE: Window B H Filter Enable. This controls H scaling filter and is effective only for non-palletized color modes. 0 = DISABLE 1 = ENABLE
6	X	B_COLOR_EXPAND: Window B 12/15/16/18-to-24 bpp color expansion. This bit should be enabled only for 12-bpp B4G4R4A4, 15-bpp B5G5R5A, 16-bpp B5G6R5, 18-bpp B6G6R6 color modes. If enabled the color conversion is performed prior to horizontal scaling. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	0x0	B_SCAN_COLUMN: Window B Scanning direction. 0= Scan in horizontal direction (for 0 or 180 degrees) 1= Scan in vertical direction (for 90 or 270 degrees) 0 = DISABLE 1 = ENABLE
2	X	B_V_DIRECTION: Window B Vertical (Y) drawing Direction 0 = INCREMENT 1 = DECREMENT
0	X	B_H_DIRECTION: Window B Horizontal (X) drawing Direction 0 = INCREMENT 1 = DECREMENT

21.11.60 DC_WIN_B_BYTE_SWAP_0

Window B Byte Swap

Offset: 0x701 │ Byte Offset: 0x1c04 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2:0	X	B_BYTE_SWAP: Window B Byte Swap. This controls byte swap of frame data read from memory prior to any data processing in the display module. 000= no byte swap (3 2 1 0) 001= byte swap for each 2-byte word (2 3 0 1) 010= byte swap for each 4-byte word (0 1 2 3) 011= word swap for each 4-byte word (1 0 3 2) 100= byte0 swapped with byte2 (3 0 1 2) 101= left shift every byte (2 1 0 3) 0 = NOSWAP 1 = SWAP2 2 = SWAP4 3 = SWAP4HW 4 = SWAP02 5 = SWAPLEFT

DC_WIN_B_COLOR_DEPTH_0

Window B Color Depth for YCbCr data format. Cb and Cr are 8-bit unsigned values. For YUV data format, U and V are 8-bit signed values. YCbCr422R is similar to YCbCr422P, but the Cb and Cr are shared vertically.

YUV422R is similar to YUV422P, but the U and V are shared vertically. YCbCr422RA is the same as YCbCr422R in memory, and YUV422RA is the same as YUV422R in memory but while reading from memory, for YCbCr422RA and YUV422RA, chroma averaging is applied for each pixel pair so that they can be processed as YUV422 by the display pipeline.

For YCbCr422R and YUV422R, every other chroma pixel is not used (discarded) by the display pipeline. B6x2G6x2R6x2A8 is similar to B8G8R6A8 but with the 2 LSBs zeroed out. R6x2G6x2B6x2A8 is similar to R8G8B6A8 but with the 2 LSBs zeroed out.

Some formats have NVCF_ or T_ prefix aliases for NVIDIA surface format naming convention, wherein left-to-right component names correspond to MSB to LSB bits within a word. In old names without NVCF_ or T_, left-to-right component names generally map LSB to MSB within a word.

Offset: 0x703 │ Byte Offset: 0x1c0c │ Read/Write: R/W │ Reset: 0x0000000c (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0001100)

Bit	Reset	Description
6:0	B8G8R8A8	B_COLOR_DEPTH: Window B Color Depth. Supported color depths are: P8 = 8-bpp (palletized) B4G4R4A4 = 12-bpp B4G4R4 B5G5R5A = 15-bpp B5G5R5

Bit	Reset	Description
		AB5G5R5 = 15-bpp B5G5R5 B5G6R5 = 16-bpp B5G6R5 B8G8R8A8 = 32-bpp B8G8R8A8 R8G8B8A8 = 32-bpp R8G8B8A8 YCbCr422 = 16-bpp YCbCr422 packed YUV422 = 16-bpp YUV422 YCbCr420P = 16-bpp YCbCr420 planar YUV420P = 16-bpp YUV420 planar YCbCr422P = 16-bpp YCbCr422 planar YUV422P = 16-bpp YUV422 planar YCbCr422R = 16-bpp YCbCr422 rotated planar YUV422R = 16-bpp YUV422 rotated planar YCbCr422RA= 16-bpp YCbCr422 rotated planar with chroma averaging YUV422RA = 16-bpp YUV422 rotated planar with chroma averaging Reserved values: 0,1,2,14,15,24,25. 3 = T_P8, P8 4 = T_A4R4G4B4, B4G4R4A4 5 = T_A1R5G5B5, B5G5R5A 6 = T_R5G6B5, B5G6R5 7 = T_R5G5B5A1, AB5G5R5 12 = T_A8R8G8B8, B8G8R8A8 13 = T_A8B8G8R8, R8G8B8A8 16 = T_U8_Y8__V8_Y8, CbYCrY422, YCbCr422 17 = T_U8_Y8__V8_Y8_TRUE, UYVY422, YUV422 18 = T_Y8___U8___V8_N420, YCbCr420P 19 = T_Y8___U8___V8_N420_TRUE, YUV420P 20 = T_Y8___U8___V8_N422, YCbCr422P 21 = T_Y8___U8___V8_N422_TRUE, YUV422P 22 = T_Y8___U8___V8_N422R, YCbCr422RP, YCbCr422R 23 = T_Y8___U8___V8_N422R_TRUE, YUV422RP, YUV422R 24 = T_V8_Y8__U8_Y8, CrYCbY422 25 = T_V8_Y8__U8_Y8_TRUE, VYUY422 26 = T_Y8, Y8 27 = T_A4B4G4R4, R4G4B4A4 28 = T_A1B5G5R5, R5G5B5A 29 = T_B5G5R5A1, AR5G5B5 30 = T_X1R5G5B5, B5G5R5X1 31 = T_R5G5B5X1, X1B5G5R5 32 = T_X1B5G5R5, R5G5B5X1 33 = T_B5G5R5X1, X1R5G5B5 34 = T_B5G6R5, R5G6B5 35 = T_B8G8R8A8 36 = T_R8G8B8A8 37 = T_X8R8G8B8, B8G8R8X8 38 = T_X8B8G8R8, R8G8B8X8 39 = T_A8Y8U8V8 40 = T_V8U8Y8A8 41 = T_Y8___U8___V8_N444, YCbCr444P 42 = T_Y8___U8V8_N420, YCrCb420SP 43 = T_Y8___V8U8_N420, YCbCr420SP 44 = T_Y8___U8V8_N422, YCrCb422SP 45 = T_Y8___V8U8_N422, YCbCr422SP 46 = T_Y8___U8V8_N422R, YCrCb422RSP 47 = T_Y8___V8U8_N422R, YCbCr422RSP 48 = T_Y8___U8V8_N444, YCrCb444SP 49 = T_Y8___V8U8_N444, YCbCr444SP 50 = T_A8Y8U8V8_TRUE 51 = T_V8U8Y8A8_TRUE 52 = T_Y8___U8___V8_N444_TRUE, YUV444P 53 = T_Y8___U8V8_N420_TRUE, YVU420SP 54 = T_Y8___V8U8_N420_TRUE, YUV420SP 55 = T_Y8___U8V8_N422_TRUE, YVU422SP 56 = T_Y8___V8U8_N422_TRUE, YUV422SP

Bit	Reset	Description
		57 = T_Y8___U8V8_N422R_TRUE, YVU422RSP 58 = T_Y8___V8U8_N422R_TRUE, YUV422RSP 59 = T_Y8___U8V8_N444_TRUE, YVU444SP 60 = T_Y8___V8U8_N444_TRUE, YUV444SP 61 = T_Y8_U8__Y8_V8, YCbYCr422 62 = T_Y8_U8__Y8_V8_TRUE, YUYV422 63 = T_Y8_V8__Y8_U8, YCrYCb422 64 = T_Y8_V8__Y8_U8_TRUE, YVYU422 65 = T_R8G8B8X8 66 = T_B8G8R8X8: new formats, reserved for future use

21.11.61 DC_WIN_B_POSITION_0

Window B Position

This register defines H position and size of Window B after scaling (if there is any)

Offset: 0x704 │ Byte Offset: 0x1c10 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	B_V_POSITION: Window B V Position. This is specified with respect to the top edge of active display area.
12:0	X	B_H_POSITION: Window B H Position. This is specified with respect to the left edge of active display area.

21.11.62 DC_WIN_B_SIZE_0

Window B Size

This register defines V position and size of Window B after scaling (if there is any)

Note: programming on window size should guarantee the whole window is inside the active area; otherwise extra rows/columns will be fetched and discarded, affecting performance.

Offset: 0x705 │ Byte Offset: 0x1c14 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	B_V_SIZE: Window B V Size (lines). This is the vertical size after scaling.
12:0	X	B_H_SIZE: Window B H Size (pixels). This is the horizontal size after scaling.

21.11.63 DC_WIN_B_PRESCALED_SIZE_0

Window B Pre-scaled Size

This register defines Window B pre-scaled size.

The H pre-scaled size is needed to determine how many bytes to fetch from memory per line and this parameter must be programmed exactly as needed taking into account the scaling factor. For planar YUV or YCbCr data formats, this parameter refer to the H pre-scaled of the Y plane.

The total number of lines to be fetched from memory is determined by post-scale V size but V pre-scaled size is needed to 'clamp' the last valid line if the vertical DDA is exactly or slightly beyond the specified V pre-scaled size.

H pre-scaled size ideally should be in terms of pixel but then hardware needs to convert this precisely to bytes to determine the amount of data to request from memory.

Offset: 0x706 │ Byte Offset: 0x1c18 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	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.

21.11.64 DC_WIN_B_H_INITIAL_DDA_0

Window B H Initial DDA

The first pixel of pre-scaled image is always used to output the first pixel so essentially this is the same as forcing the H Initial DDA integer portion to 1 initially even though user typically programs this to 0. If it makes the implementation easier, it is possible to force software to program the Initial DDA integer portion to 1. Similarly with the V Initial DDA.

Offset: 0x707 │ Byte Offset: 0x1c1c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	B_H_INITIAL_DDA: Window B H Initial DDA (4.12). This is typically programmed to 0.0

21.11.65 DC_WIN_B_V_INITIAL_DDA_0

Window B V Initial DDA

Offset: 0x708 │ Byte Offset: 0x1c20 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	B_V_INITIAL_DDA: Window B V Initial DDA (4.12). This is typically programmed to 0.0 for both non-interlaced and interlaced sources.

21.11.66 DC_WIN_B_DDA_INCREMENT_0

Window B DDA Increment

The DDA increment is typically calculated by dividing (Pre-scaled size in pixels - 1) by (Post-scaled size in pixels - 1). The result should be rounded up and expressed as 4.12 format (4-bit integer and 12-bit fraction). For non-filtered images, this value can be slightly larger so that it is not missing the last row/column. Reference programming values (H and V should be calculated separately depending on filter on/off and sizes):

- Filter on: min(round((prescaled_size_in_pixels - 1) * 0x1000 / (post_scaled_size_in_pixels - 1)) + initial_dda_bias, MAX)
- Filter off: min(round(prescaled_size_in_pixels * 0x1000 / (post_scaled_size_in_pixels - 1) - 0.5) + initial_dda_bias, 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/pixel formats.

8.0 (0x8000) for 2 Bytes/pixel formats.

Where the value of initial_dda_bias is as follows:

For V_DDA_INCREMENT: initial_dda_bias = (v_intial_dda == negative) ? mod(v_initial_dda)/(post_scaled_size_in_pixels - 1) : 0 ;

For H_DDA_INCREMENT: initial_dda_bias = 0;

They are theoretically the biggest values that guarantee not displaying beyond an image boundary.

If the DDA increment is less than 1.0 then the image will be upscaled.If the DDA increment is more than 1.0 then the image will be downscaled.

Offset: 0x709 │ Byte Offset: 0x1c24 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	B_V_DDA_INCREMENT: Window B Vertical DDA Increment (4.12) This should be set to 1.0 if there is no scaling. Maximum value is 15.0 regardless of the number of bytes per pixel.
15:0	X	B_H_DDA_INCREMENT: Window B Horizontal DDA Increment (4.12) This should be set to 1.0 if there is no scaling. The maximum value for downscaling depends on the number of bytes per pixel. For 4-byte/pixel modes (32-bpp) the maximum value is 4.0 and for all other modes the maximum value is 8.0.

21.11.67 DC_WIN_B_LINE_STRIDE_0

Window B Line Stride

Offset: 0x70a │ Byte Offset: 0x1c28 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	B_UV_LINE_STRIDE: Window B Line Stride for Chroma This is stride (in bytes) for planar YUV or YCbCr data formats for the chroma plane, with the restriction that it must be programmed to be multiples of 4 (16 if tiled or in horizontal flipping) This is not used (ignored) for other non-planar data formats.
15:0	X	B_LINE_STRIDE: Window B Line Stride This is stride (in bytes) for all non-planar data formats. If the memory surface is tiled, the stride needs to be a multiple of 16. If H_DIRECTION of window B is set to DECREMENT, the stride also needs to be a multiple of 16. For planar YUV or YCbCr data formats, this is stride (in bytes) for the luma plane with the restriction that it must be multiples of 8 (16 if tiled or in horizontal flipping). For tiled surface, this value may affect starting address of a window. Refer to the comment of START_ADDR for detail.

21.11.68 DC_WIN_B_DV_CONTROL_0

Window B Digital Vibrance Control

If enabled, Digital Vibrance is applied after H and V scaling and after color palette or color space conversion logic but before color keying multiplexer and before cursor multiplexer.

- After DV, new R = R + (2R - G - B) * FR, where FR is fraction from 0 to 7/8
- After DV, new G = G + (2G - R - B) * FG, where FG is fraction from 0 to 7/8
- After DV, new B = B + (2B - R - G) * FB, where FB is fraction from 0 to 7/8

Offset: 0x70e │ Byte Offset: 0x1c38 │ Read/Write: R/W │ Reset: 0x000X0X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18:16	X	B_DV_CONTROL_B: Digital Vibrance control for B
10:8	X	B_DV_CONTROL_G: Digital Vibrance control for G
2:0	X	B_DV_CONTROL_R: Digital Vibrance control for R

Class: Display Color Keying and Blending

Color keying and blending of the display windows are done prior to cursor blending.

Cursor always goes on top of the blended windows. Blending is controlled independently on each possible overlap area of the display windows. If 3 windows are enabled there are 7 possible overlap area combinations. For every window in each overlap area combination, color key can be disabled or enabled. Also, for every window in each overlap area combination, there is a corresponding window blend control parameter and a window blend weight parameter. The window blend control parameter is always effective but the window blend weight is not always used. The window blend weight can also be derived from pixel alpha value or from the reverse of other overlapping windows weight.

Color keying has the highest priority for display window blending. Color key consists of a range of color which is searched independently for each windows. If more than 1 windows color key is enabled, then Window A color key has the highest priority, followed by Window B color key, and then followed by Window C color key. Two sets of color key ranges (Color Key 0 and Color Key 1) can be defined; they are shared for all windows. It is possible to use both color key sets for the same window or for two separate windows.

The two sets of color key ranges should not overlap. If they do, the overlapped colors are treated as if they are part of Color Key 0 and not part of Color Key 1.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is enabled for any overlap condition and the window pixel is not within the color key range (key not match), then the window pixel will not be blended with other overlapping window pixels but it will be weighted. The weight is not dependent on the overlap condition it is controlled by the same set of parameters for all overlapping condition.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is enabled for a particular overlap condition and the window pixel is within the color key range (key match), then the window pixel will be weighted and blended with other overlapping pixels and this is controlled separately for each overlap condition.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is disabled then the window pixel will be blended with other overlapping pixels and this is controlled separately for each overlap condition.

Display Color Key Parameters

For B4G4R4A4, B5G6R5A, B5G6R5 modes, the color key should be compared prior to color conversion to 24-bpp and unused least significant bits of the pixel are filled with zeros.

For palletized mode, the color key is compared prior to the color palette, and the palletized color is compared against the green color key values/mask.

For YUV mode, U and V are offset by +128 before performing the color key comparison.

In all cases, the color key is compared prior to the horizontal/vertical scaling filter and prior to digital vibrance control. Both upper and lower values are inclusive.

21.11.69 DC_WIN_B_BLEND_LAYER_CONTROL_0

Window B

Offset: 0x716 | Byte Offset: 0x1c58 | Read/Write: R/W | Reset: 0x01000000 (0bxxxx00010000000000000000000000)

Bit	Reset	Description
27:25	0x0	B_COLOR_KEY_SELECT: 0 = NONE 1 = WINDOWA_KEY0 2 = WINDOWA_KEY1 3 = WINDOWB_KEY0 4 = WINDOWB_KEY1 5 = WINDOWC_KEY0 6 = WINDOWC_KEY1
24	BLEND_BYPASS	B_BLEND_BYPASS: 0 = BLEND_ENABLE 1 = BLEND_BYPASS
23:16	0x0	B_K2
15:8	0x0	B_K1
7:0	0x0	B_WINDOW_LAYER_DEPTH

21.11.70 DC_WIN_B_BLEND_MATCH_SELECT_0

Offset: 0x717 | Byte Offset: 0x1c5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx00xx00x000x000)

Bit	Reset	Description
13:12	0x0	B_BLEND_FACTOR_DST_ALPHA_MATCH_SELECT: 0 = ZERO 1 = ONE 2 = NEG_K1_TIMES_SRC 3 = K2
9:8	0x0	B_BLEND_FACTOR_SRC_ALPHA_MATCH_SELECT: 0 = ZERO 1 = K1 2 = K2
6:4	0x0	B_BLEND_FACTOR_DST_COLOR_MATCH_SELECT: 0 = ZERO 1 = ONE 2 = K1 3 = K2 4 = K1_TIMES_DST 5 = NEG_K1_TIMES_DST 6 = NEG_K1_TIMES_SRC 7 = NEG_K1
2:0	0x0	B_BLEND_FACTOR_SRC_COLOR_MATCH_SELECT: 0 = ZERO 1 = ONE 2 = K1 3 = K1_TIMES_DST 4 = NEG_K1_TIMES_DST 5 = K1_TIMES_SRC

21.11.71 DC_WIN_B_BLEND_NOMATCH_SELECT_0

Offset: 0x718 | Byte Offset: 0x1c60 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xx00x000x000)

Bit	Reset	Description
13:12	0x0	B_BLEND_FACTOR_DST_ALPHA_NOMATCH_SELECT: 0 = ZERO 1 = ONE 2 = NEG_K1_TIMES_SRC 3 = K2
9:8	0x0	B_BLEND_FACTOR_SRC_ALPHA_NOMATCH_SELECT: 0 = ZERO 1 = K1 2 = K2
6:4	0x0	B_BLEND_FACTOR_DST_COLOR_NOMATCH_SELECT: 0 = ZERO 1 = ONE 2 = K1 3 = K2 4 = K1_TIMES_DST 5 = NEG_K1_TIMES_DST 6 = NEG_K1_TIMES_SRC 7 = NEG_K1
2:0	0x0	B_BLEND_FACTOR_SRC_COLOR_NOMATCH_SELECT: 0 = ZERO 1 = ONE 2 = K1 3 = K1_TIMES_DST 4 = NEG_K1_TIMES_DST 5 = K1_TIMES_SRC

21.11.72 DC_WIN_B_BLEND_ALPHA_1BIT_0

Offset: 0x719 | Byte Offset: 0x1c64 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	B_BLEND_WEIGHT1: This is used only for 1-bit alpha and alpha value of 1.
7:0	0x0	B_BLEND_WEIGHT0: This is used only for 1-bit alpha and alpha value of 0.

21.12 WINBUF_B Registers

The registers under DC_WINBUF are triple-buffered.

21.12.1 DC_WINBUF_B_START_ADDR_0

Window B Start Address

Overview

START_ADDR, BUF_STRIDE, LINE_STRIDE, ADDR_H_OFFSET, ADDR_V_OFFSET combined, specify the starting address of a window buffer in the memory surface. Generally START_ADDR is programmed with the starting address of the memory surface. H/V_OFFSET specifies the address offsets of the pixel at the beginning of the window, with respect to the starting address of the memory surface. (There is an exception to that when memory surface is not well aligned, see ”For linear address mode” in Programming Restrictions below.)

Note that "beginning of the window" here means the top-left corner of a window in normal mode, top-right corner in horizontal flipping, bottom-left corner in vertical flipping, and bottom-right in horizontal+vertical flipping.

Starting Address Calculation

These formulae are the same for both tiled and linear modes. However, for linear mode, the addresses calculated are real physical addresses, but for tiled mode, these addresses will be translated to the real physical addresses by the memory controller client before used.

- When a window is host triggered, the starting address of a window is calculated as follows by hardware:
  - non-YUV-planar modes:

    starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - YUV-planar modes:

    Y-starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    U-starting-address = START_ADDR_U + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    V-starting-address = START_ADDR_V + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by hardware.
- When a window is non-host triggered, starting address of a window buffer is calculated as below:
  - non-YUV-planar mode:

    starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - YUV-planar mode:

    Y-starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    U-starting-address = START_ADDR_U + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    V-starting-address = START_ADDR_V + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by hardware.

    buf_index is the index transmitted by the triggering module pointing to the first buffer of a frame.

Programming Restrictions

- For tiled address mode:

  Image surface can only aligned to multiples of 256, thus the following restrictions.
  - START_ADDR, START_ADDR_U, START_ADDR_V need to be multiples of 256.
  - BUF_STRIDE, UV_BUF_STRIDE need to be multiples of 256
  - LINE_STRIDE, UV_LINE_STRIDE need to be multiples of 16
  - ADDR_H_OFFSET needs to be even in YUV planar format, or a multiple of bytes per pixel in other formats. If H_DIRECTION=DECREMENT, however, it should point to last valid byte, which is an odd offset.
  - ADDR_V_OFFSET needs to be multiple of 2 in YUV planar format, unless H_DIRECTION=DECREMENT, but with no restrictions on other color formats.
- For linear address mode:

  Image surface can be aligned 2, 4 or 8 bytes, depending on the color formats.

  As an additional restriction for display, START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16.

When a surface is not aligned to 16 bytes, program START_ADDR with the memory surface address with its least 4 significant bits zeroed out and add these 4 address bits to the original H_OFFSET. (So the formulae in Starting Address Calculation above still hold)

- For all formats:

  START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16.
- For 16-bpp formats,

  (START_ADDR+H_OFFSET) needs to be a multiple of 2.
- For 32-bpp formats,

  (START_ADDR+H_OFFSET) needs to be a multiple of 4.
- For YUV planar formats:

  BUF_STRIDE, UV_BUF_STRIDE:

  BUF_STRIDE[2:1]=UV_BUF_STRIDE[1:0]

  or as a stricter constraint: BUF_STRIDE be multiple of 8, UV_BUF_STRIDE be a multiple of 4.

  LINE_STRIDE, UV_LINE_STRIDE:

  LINE_STRIDE and UV_LINE_STRIDE need to be at least 16.

  LINE_STRIDE needs to be multiple of 8, UV_LINE_STRIDE needs to be a multiple of 4.

  ADDR_H_OFFSET: Needs to be even unless H_DIRECTION=DECREMENT, in which case should point to the last byte pixel, which is at an odd offset.

  ADDR_V_OFFSET: Needs to be even unless V_DIRECTION=DECREMENT, in which case should point to the last valid line, which is at an odd offset.

- Memory allocation for non-host triggered case

  When a window buffer is not controlled by host (software) then a frame may be stored in multiple buffers. In this case, the buffers must be contiguous in the memory because the display will use the same luma or chroma line strides for all lines in the frame.

  Also buffer wraparound must not occur in the middle of the displayed part of the frame. The controlling module will send frame start and frame end indicators (flags) to display module to indicate the beginning and end of frame. Buffer start address is latched when frame start flag is active but the actual buffer start address is not switched until frame end flag is active. In the case where one buffer corresponds to one frame then frame start and frame end flag are active every time a buffer index is sent.

Offset: 0x800 │ Byte Offset: 0x2000 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_START_ADDR: Window B Start Address. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

21.12.2 DC_WINBUF_B_START_ADDR_NS_0

Window B Shadowed Start Address

Offset: 0x801 │ Byte Offset: 0x2004 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_START_ADDR_NS: Window B Shadowed Start Address. This is ARM set shadow of Start Address.

21.12.3 DC_WINBUF_B_START_ADDR_U_0

Window B Start Address for U plane

Offset: 0x802 | Byte Offset: 0x2008 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_START_ADDR_U: Window B Start Address for U plane. This is a byte address.

21.12.4 DC_WINBUF_B_START_ADDR_U_NS_0

Window B Shadowed Start Address for U plane

Offset: 0x803 | Byte Offset: 0x200c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_START_ADDR_U_NS: Window B Shadowed Start Address for U plane. This is ARM set shadow register of U start address

21.12.5 DC_WINBUF_B_START_ADDR_V_0

Window B Start Address for V plane

Offset: 0x804 | Byte Offset: 0x2010 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_START_ADDR_V: Window B Start Address for V plane. This is a byte address.

21.12.6 DC_WINBUF_B_START_ADDR_V_NS_0

Window B Shadowed Start Address for V plane

Offset: 0x805 | Byte Offset: 0x2014 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_START_ADDR_V_NS: Window B Shadowed Start Address for V plane. This is ARM set shadow register of U start address

21.12.7 DC_WINBUF_B_ADDR_H_OFFSET_0

Window B Horizontal address offset

Offset: 0x806 | Byte Offset: 0x2018 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_ADDR_H_OFFSET: Window B Horizontal address offset. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies horizontal offset of Y plane. The horizontal offsets of U/V plane is derived by hardware.

21.12.8 DC_WINBUF_B_ADDR_H_OFFSET_NS_0

Window B Shadowed Horizontal address offset

Offset: 0x807 | Byte Offset: 0x201c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_ADDR_H_OFFSET_NS: Window B Shadowed Horizontal address offset. This is ARM set shadow of ADDR_H_OFFSET

21.12.9 DC_WINBUF_B_ADDR_V_OFFSET_0

Window B Vertical address offset

Offset: 0x808 | Byte Offset: 0x2020 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_ADDR_V_OFFSET: Window B Vertical address offset. This is the vertical coordinate. For YUV planar pixel format, this specifies the vertical coordinate of Y plane. The vertical coordinate of U/V plane is derived by hardware.

21.12.10 DC_WINBUF_B_ADDR_V_OFFSET_NS_0

Window B Shadowed Vertical address offset

Offset: 0x809 | Byte Offset: 0x2024 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_ADDR_V_OFFSET_NS: Window B Shadowed Vertical address offset. This is ARM set shadow of ADDR_V_OFFSET.

21.12.11 DC_WINBUF_B_UFLOW_STATUS_0

Window B FIFO Underflow Status Register

Offset: 0x80a | Byte Offset: 0x2028 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	COUNT_OFLOW: Flag bit that indicates that the underflow event counter has overflowed. There were too many events. If COUNT_OFLOW is set, UFLOW_COUNT is meaningless. Cleared on write.
23:0	X	UFLOW_COUNT: Underflow count. This field indicates the number of contiguous groups of output pixels for which there was no data in the FIFO. For example, if the valid from the FIFO is low for 10 consecutive cycles and then goes high, the counter will increment by one. Reset to zero on write.

21.12.12 DC_WINBUF_B_SURFACE_KIND_0

Window B Surface Kind

Offset: 0x80b | Byte Offset: 0x202c | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx001xx00)

Bit	Reset	Description
6:4	0x1	B_BLOCK_HEIGHT: Block Height: For block linear, height of block in GOBs 0 = HEIGHT_1 1 = HEIGHT_2 2 = HEIGHT_4 3 = HEIGHT_8 4 = HEIGHT_16 5 = HEIGHT_32
1:0	0x0	B_SURFACE_KIND: Surface Kind: Pitched, Tiled, or Block Linear 0 = PITCH 1 = TILED 2 = BL_16B2

21.12.13 DC_WINBUF_B_SURFACE_WEIGHT_0

Window B Surface Weights

Offset: 0x80c | Byte Offset: 0x2030 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
6:5	X	B_SURFACE_WEIGHT_V: Window B V Surface Weights 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16 0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
4:3	X	B_SURFACE_WEIGHT_U: Window B U or UV Surface Weights 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16 0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
2:1	X	B_SURFACE_WEIGHT_Y: Window B Y or packed Surface Weights 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16 0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
0	0x0	B_SURFACE_WEIGHT_OVERRIDE: weight override 0 = DISABLE 1 = ENABLE

21.12.14 DC_WINBUF_B_START_ADDR_HI_0

Window B Higher 32 bits of Start Address

Make the start address 64 bits by adding the higher 32 bits below.

Offset: 0x80d │ Byte Offset: 0x2034 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00))

Bit	Reset	Description
1:0	0x0	B_START_ADDR_HI: Window B Start Address. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

21.12.15 DC_WINBUF_B_START_ADDR_HI_NS_0

Window B Shadowed Higher 32 bits of Start Address

Offset: 0x80e │ Byte Offset: 0x2038 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00))

Bit	Reset	Description
1:0	0x0	B_START_ADDR_HI_NS: Window B Shadowed Start Address. This is ARM set shadow of Start Address.

21.12.16 DC_WINBUF_B_START_ADDR_HI_U_0

Window B Higher 32 bits of Start Address for U plane

Offset: 0x80f │ Byte Offset: 0x203c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00))

Bit	Reset	Description
1:0	0x0	B_START_ADDR_HI_U: Window B Start Address for U plane. This is a byte address.

21.12.17 DC_WINBUF_B_START_ADDR_HI_U_NS_0

Window B Shadowed Higher 32 bits of Start Address for U plane

Offset: 0x810 │ Byte Offset: 0x2040 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00))

Bit	Reset	Description
1:0	0x0	B_START_ADDR_HI_U_NS: Window B Shadowed Higher 32 bits of Start Address for U plane. This is ARM set shadow register of U start address

21.12.18 DC_WINBUF_B_START_ADDR_HI_V_0

Window B Higher 32 bits of Start Address for V plane

Offset: 0x811 │ Byte Offset: 0x2044 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00))

Bit	Reset	Description
1:0	0x0	B_START_ADDR_HI_V: Window B Higher 32 bits of Start Address for V plane. This is a byte address.

21.12.19 DC_WINBUF_B_START_ADDR_HI_V_NS_0

Window B Shadowed Higher 32 bits of Start Address for V plane

Offset: 0x812 │ Byte Offset: 0x2048 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00))

Bit	Reset	Description
1:0	0x0	B_START_ADDR_HI_V_NS: Window B Shadowed Higher 32 bits of Start Address for V plane. This is ARM set shadow register of U start address

21.12.20 DC_WINBUF_B_START_ADDR_FIELD2_0

Window B Start Address

Interlace/Stereo support. 64-bit base address for the odd field/right frame.

Offset: 0x813 | Byte Offset: 0x204c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_START_ADDR_FIELD2: Window B Start Address. This is a byte address. The LSB is not used for the 16-bpp non-planar pixel format, and the last 2 LSBs are not used for the 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

21.12.21 DC_WINBUF_B_START_ADDR_FIELD2_NS_0

Window B Shadowed Start Address

Offset: 0x814 | Byte Offset: 0x2050 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_START_ADDR_FIELD2_NS: Window B Shadowed Start Address. This is the ARM set shadow of the start address.

21.12.22 DC_WINBUF_B_START_ADDR_FIELD2_U_0

Window B Start Address for U plane

Offset: 0x815 | Byte Offset: 0x2054 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_START_ADDR_FIELD2_U: Window B Start Address for U plane. This is a byte address.

21.12.23 DC_WINBUF_B_START_ADDR_FIELD2_U_NS_0

Window B Shadowed Start Address for U plane

Offset: 0x816 | Byte Offset: 0x2058 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_START_ADDR_FIELD2_U_NS: Window B Shadowed Start Address for U plane. This is the ARM set shadow register of the U start address.

21.12.24 DC_WINBUF_B_START_ADDR_FIELD2_V_0

Window B Start Address for V plane

Offset: 0x817 | Byte Offset: 0x205c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_START_ADDR_FIELD2_V: Window B Start Address for V plane. This is a byte address.

21.12.25 DC_WINBUF_B_START_ADDR_FIELD2_V_NS_0

Window B Shadowed Start Address for V plane

Offset: 0x818 | Byte Offset: 0x2060 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_START_ADDR_FIELD2_V_NS: Window B Shadowed Start Address for V plane. This is the ARM set shadow register of the U start address.

21.12.26 DC_WINBUF_B_START_ADDR_FIELD2_HI_0

Window B Higher 32 bits of Start Address

Offset: 0x819 | Byte Offset: 0x2064 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	B_START_ADDR_FIELD2_HI: Window B Start Address. This is a byte address. The LSB is not used for the 16-bpp non-planar pixel format, and the last 2 LSBs are not used for the 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies the start address for the Y plane.

21.12.27 DC_WINBUF_B_START_ADDR_FIELD2_HI_NS_0

Window B Shadowed Higher 32 bits of Start Address

Offset: 0x81a | Byte Offset: 0x2068 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	B_START_ADDR_FIELD2_HI_NS: Window B Shadowed Start Address. This is the ARM set shadow of the start address.

21.12.28 DC_WINBUF_B_START_ADDR_FIELD2_HI_U_0

Window B Higher 32 bits of Start Address for U plane

Offset: 0x81b | Byte Offset: 0x206c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	B_START_ADDR_FIELD2_HI_U: Window B Start Address for U plane. This is a byte address.

21.12.29 DC_WINBUF_B_START_ADDR_FIELD2_HI_U_NS_0

Window B Shadowed Higher 32 bits of Start Address for U plane

Offset: 0x81c | Byte Offset: 0x2070 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	B_START_ADDR_FIELD2_HI_U_NS: Window B Shadowed Higher 32 bits of Start Address for U plane. This is the ARM set shadow register of the U start address.

21.12.30 DC_WINBUF_B_START_ADDR_FIELD2_HI_V_0

Window B Higher 32 bits of Start Address for V plane

Offset: 0x81d | Byte Offset: 0x2074 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	B_START_ADDR_FIELD2_HI_V: Window B Higher 32 bits of Start Address for V plane. This is a byte address.

21.12.31 DC_WINBUF_B_START_ADDR_FIELD2_HI_V_NS_0

Window B Shadowed Higher 32 bits of Start Address for V plane

Offset: 0x81e | Byte Offset: 0x2078 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	B_START_ADDR_FIELD2_HI_V_NS: Window B Shadowed Higher 32 bits of Start Address for V plane. This is the ARM set shadow register of the U start address.

21.12.32 DC_WINBUF_B_ADDR_H_OFFSET_FIELD2_0

Window B Horizontal address offset

Offset: 0x81f | Byte Offset: 0x207c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_ADDR_H_OFFSET_FIELD2: Window B Horizontal address offset. This is a byte address. The LSB is not used for the 16-bpp non-planar pixel format, and the last 2 LSBs are not used for the 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies the horizontal offset of the Y plane. The horizontal offsets of U/V plane are derived by hardware.

21.12.33 DC_WINBUF_B_ADDR_H_OFFSET_FIELD2_NS_0

Window B Shadowed Horizontal address offset

Offset: 0x820 | Byte Offset: 0x2080 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_ADDR_H_OFFSET_FIELD2_NS: Window B Shadowed Horizontal address offset. This is the ARM set shadow of ADDR_H_OFFSET.

21.12.34 DC_WINBUF_B_ADDR_V_OFFSET_FIELD2_0

Window B Vertical address offset

Offset: 0x821 | Byte Offset: 0x2084 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_ADDR_V_OFFSET_FIELD2: Window B Vertical address offset. This is the vertical coordinate. For YUV planar pixel format, this specifies the vertical coordinate of the Y plane. The vertical coordinate of U/V plane is derived by hardware.

21.12.35 DC_WINBUF_B_ADDR_V_OFFSET_FIELD2_NS_0

Window B Shadowed Vertical address offset

Offset: 0x822 | Byte Offset: 0x2088 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_ADDR_V_OFFSET_FIELD2_NS: Window B Shadowed Vertical address offset. This is the ARM set shadow of ADDR_V_OFFSET

21.12.36 DC_WINBUF_B_UFLOW_CTRL_0

When the underflow debug mode is enabled. Instead of sending out the last valid pixel, DBG_PIXEL is sent out.

Offset: 0x824 | Byte Offset: 0x2090 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	B_UFLOW_CTRL_DBG_MODE: 0 = DISABLE 1 = ENABLE

21.12.37 DC_WINBUF_B_UFLOW_DBG_PIXEL_0

Offset: 0x825 | Byte Offset: 0x2094 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_UFLOW_DBG_PIXEL

21.12.38 DC_WINBUF_B_UFLOW_THRESHOLD_0

Offset: 0x826 | Byte Offset: 0x2098 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxx0000000000000)

Bit	Reset	Description
12:0	0x0	B_UFLOW_THRESHOLD

21.12.39 DC_WINBUF_B_SPOOL_UP_0

Spool up time configuration for different windows related logic.

SPOOL_UP_CTRL = MAX - This is the current Tegra K1 behavior. This allows for memfetch to start fetching for a window (irrespective of its relative position on the screen), at the very beginning of the vblank of the next frame. Since the spool up time is maximum, this should have the least probability of underflow of the display pipe. However, on the down side, since all three windows would start fetching at the same time, and since display is a high priority client for MC, it has a high potential to starve out other clients.

SPOOL_UP_CTRL = PROGRAMMABLE, SPOOL_UP_DURATION = 1 - This corresponds to the Tegra 3 behavior. Since the fetch starts only 1 line prior to the active scan line of a window, this has high probability of causing underflow, but has a lesser impact on the other MC clients.

So, this register programmability programs us a method to modulate the spool up time between the best and worst, if required, when other MC clients are present. Let's say when GPU/VIC is active, we may not want a very aggressive spool up and program some definite spool up duration. So, spool up time is a general knob, which would be a function of the resolution, bpp, active clients, memory speed, etc.

SPOOL_UP_EDGE:- By default the memfetch start signal generated by the hvtgen logic is one line clock wide.

For SPOOL_UP_CTRL = MAX:

- In this case if SPOOL_UP_EDGE is programmed to NEGEDGE. This ensures that if any act_req is sent during the time vcounter = 0(at the beginning of the frame). The activate value is loaded into the active state and the memfetch would see the latest updated value. The downside of programming it to NEGDGE is that 1 line clock worth of spool up is lost. This is required based upon Tegra 3 compatibility and the way tests are written.
- In this case if SPOOL_UP_EDGE is programmed to POSEDGE. Then the memfetch would start fetching at the very beginning of the frame start pulse (1 line clock wide) and would give maximum spool up time. However, this may give test failure and potential Tegra 3 compatibility issues as described above

FOR SPOOL_UP_CTRL = PROGRAMMABLE:

- Special care must be taken here. If SPOOL_UP_DURATION is set to 1. SPOOL_UP_EDGE must be set to POSEDGE to allow for at least 1 line worth of spool up.
- If SPOOL_UP_DURATION is > 1. Then SPOOL_UP_EDGE can be either POSEDGE or NEGEDGE. If its NEGEDGE, then the effective spool up duration would be 1 line clock less than the SPOOL_UP_DURATION.

Offset: 0x827 │ Byte Offset: 0x209c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
28:16	X	B_SPOOL_UP_DURATION
1	0x0	B_SPOOL_UP_EDGE: 0 = NEGEDGE 1 = POSEDGE
0	0x0	B_SPOOL_UP_CTRL: 0 = MAX 1 = PROGRAMMABLE

21.12.40 DC_WINBUF_B_SCALEFACTOR_THRESHOLD_0

Registers for latency allowance scaling. Threshold is in terms of 64 byte atoms. Display memfetch compares its buffer occupancy with the threshold and generates indication to MC if its buffer occupancy is above HWM or below LWM.

Offset: 0x828 │ Byte Offset: 0x20a0 │ Read/Write: R/W │ Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:16	0xffff	B_SF_LWM_THRESHOLD
15:0	0xffff	B_SF_HWM_THRESHOLD

21.12.41 DC_WINBUF_B_LATENCY_THRESHOLD_0

The LATENCY_THRESHOLD register controls the behavior of the rdy4_latency_event from the display to the MC.

If B_RDY4LATENCY_THRESHOLD_ENABLE is set to DISABLE, the sideband rdy4_latency_event is always asserted, indicating that the display is always ready for a DVFS event.

The B_RDY4LATENCY_THRESHOLD field indicates the required occupancy of line buffers of windows below which the sideband rdy4_latency_event is deasserted. The occupancy figure is in units of 64B atoms for memfetch windows and in units of scan lines for regular windows.

The B_RDY4LATENCY_SPOOLUP_DURATION field indicates the number of scan lines during spool-up when the sideband will remain asserted (DVFS is allowed). This takes affect only when B_RDY4LATENCY_SPOOLUP_CTRL is set to ALLOW.

If the B_RDY4LATENCY_SPOOLUP_CTRL field is set to DISALLOW, the above register field is not affected and DVFS is disallowed until the threshold is reached.

Offset: 0x829 │ Byte Offset: 0x20a4 │ Read/Write: R/W │ Reset: 0x0000ffff (0b00x00000000000001111111111111111)

Bit	Reset	Description
31	DISABLE	B_RDY4LATENCY_THRESHOLD_ENABLE: 0 = DISABLE 1 = ENABLE
30	DISALLOW	B_RDY4LATENCY_SPOOLUP_CTRL: 0 = DISALLOW 1 = ALLOW
28:16	0x0	B_RDY4LATENCY_SPOOLUP_DURATION
15:0	0xffff	B_RDY4LATENCY_THRESHOLD

21.12.42 DC_WINBUF_B_MEMFETCH_DEBUG_STATUS_0

Memfetch Debug Status

Offset: 0x82a │ Byte Offset: 0x20a8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000000000000)

Bit	Reset	Description
16	0x0	B_DEBUG_ENABLE:reg to enable FGCG for flops added to debug signals in memfetch.
15	0x0	B_ALIGN_FIFO_NON_IDLE: align FIFO idle at end of frame. 0= align FIFO is idle at end of frame. 1= align FIFO is not idle at end of frame
14	0x0	B_PIPE1_FIFO_NON_IDLE: pipe1 FIFO idle at end of frame. 0= pipe1 FIFO is idle at end of frame. 1= pipe1 FIFO is not idle at end of frame
13	0x0	B_PIPE0_FIFO_NON_IDLE: pipe0 FIFO idle at end of frame. 0= pipe0 FIFO is idle at end of frame. 1= pipe0 FIFO is not idle at end of frame
12	0x0	B_FRAME_FIFO_NON_IDLE: frame FIFO idle at end of frame. 0= frame FIFO is idle at end of frame. 1= frame FIFO is not idle at end of frame
11	0x0	B_SURFACE_FIFO_NON_IDLE: surface FIFO idle at end of frame. 0= surface FIFO is idle at end of frame. 1= surface FIFO is not idle at end of frame
10	0x0	B_TAG_FIFO_NON_IDLE: tag FIFO idle at end of frame. 0= tag FIFO is idle at end of frame. 1= tag FIFO is not idle at end of frame
9	0x0	B_RSP_FIFO_NON_IDLE: Response FIFO idle at end of frame. 0= response FIFO is idle at end of frame. 1= response FIFO is not idle at end of frame
8	0x0	B_REQ_FIFO_NON_IDLE: Request FIFO idle at end of frame. 0= request FIFO is idle at end of frame. 1= request FIFO is not idle at end of frame
7	0x0	B_DP_VALID_NON_IDLE: DP lines valid at end of frame. 0= DP lines are not valid at end of frame. 1= DP lines are valid at end of frame
6	0x0	B_V_SSM_NON_IDLE: V SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame
5	0x0	B_U_SSM_NON_IDLE: U SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame

Bit	Reset	Description
4	0x0	B_Y_SSM_NON_IDLE: Y SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame
3	0x0	B_DP_POOL_AVAIL: DP thinks there is data in the pool at end of frame. 0= No pool data available at end of frame. 1= pool data available at end of frame
2	0x0	B_POOL_NOT_EMPTY: Status of pool at end of frame. 0= pool is empty at end of frame. 1= pool is not empty at end of frame
1	0x0	B_UNDERFLOW_LINE1: Underflow of line0. 0= No underflow 1= line0 underflowed
0	0x0	B_UNDERFLOW_LINE0: Underflow of line0. 0= No underflow 1= line0 underflowed

21.12.43 DC_WINBUF_B_MEMFETCH_CONTROL_0

Note: MEMFETCH_CLK_GATE_OVERRIDE should only be enabled AFTER the vertical scaler settings in the active copy - VDDA_INCREMENT is properly set up.

Memfetch Control Register

Offset: 0x82b │ Byte Offset: 0x20ac │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00) | Default: 0x00000001

Bit	Reset	SW Default	Description
1	0x0	NONE	B_MEMFETCH_CLK_GATE_OVERRIDE: disable clock-gating of memfetch for a controller (all three windows) 0 = DISABLE 1 = ENABLE
0	0x0	ENABLE	B_MEMFETCH_RESET: reset memfetch for a controller (all three windows) 0 = DISABLE 1 = ENABLE

21.12.44 DC_WINBUF_B_OCCUPANCY_THROTTLE_0

Offset: 0x82c │ Byte Offset: 0x20b0 │ Read/Write: R/W │ Reset: 0xffff0000 (0b1111111111111111xxxxxxxxxxxxxxx0)

Bit	Reset	Description
31:16	0xffff	B_OCCUPANCY_MAX_THRESHOLD
0	0x0	B_OCCUPANCY_THROTTLE_MODE: 0 = DISABLE 1 = ENABLE

21.12.45 DC_WINBUF_B_SCRATCH_REGISTER_0_0

Offset: 0x82d │ Byte Offset: 0x20b4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	B_SCRATCH_REGISTER_0:Scratch register 0

21.12.46 DC_WINBUF_B_SCRATCH_REGISTER_1_0

Offset: 0x82e │ Byte Offset: 0x20b8 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	B_SCRATCH_REGISTER_1:Scratch register 1

21.13 Window C (WIN_C) Registers

These registers control window C parameters.

21.13.1 DC_WINC_C_COLOR_PALETTE_0

Window C Color Palette

This is used for palletized data format (color depth of 8-bpp or less) or for gamma correction for non-palletized data formats (color depth of more than 8-bpp). Each window has its own color palette which consists of three 256x8 register file which can be written by host and indexed (read) by the window.

For palletized data format less than 8-bpp, the pixel data is aligned to least significant bits of the palette index (address) and the remaining upper bits are filled with the corresponding bits of the Palette Color Extension. For example, for 4-bpp mode, the pixel data occupies bits 3:0 of the palette index and bits 7:4 of the palette index are set to bits 7:4 of the Palette Color Extension. Host read is assumed to be not needed - software can cache the color palette in system memory. This is an array of 256 identical register entries; the register fields below apply to each entry.

Color Palette

Offset: 0x500..0x5ff │ Byte Offset: 0x1400..0x17fc │ Read/Write: WO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	C_COLOR_PALETTE_B: Blue Color Palette
15:8	X	C_COLOR_PALETTE_G: Green Color Palette
7:0	X	C_COLOR_PALETTE_R: Red Color Palette

21.13.2 DC_WINC_C_PALETTE_COLOR_EXT_0

Window C Palette Color Extension

Palette extension for 1-bpp, 2-bpp, and 4-bpp. These bits provide the upper most significant bits for indexing the color palette.

Offset: 0x600 │ Byte Offset: 0x1800 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:1	X	C_PALETTE_COLOR_EXT: Window C Palette Color Extension. Bits 7:1 are used for 1-bpp mode, bits 7:2 are used for 2-bpp mode, and bits 7:4 are used for 4-bpp mode

21.13.3 DC_WINC_C_H_FILTER_P00_0

Window C Horizontal Filter phase 00

Horizontal scaling filter coefficients.

Horizontal scaling filter is a 6-tap filter with 4-bit positional phase.

- Coefficients 0 and 5 are 3-bit signed values ranging from -4 to 3.
- Coefficients 1 and 4 are 5-bit signed values ranging from -16 to 15.
- Coefficients 2 and 3 are 8-bit unsigned values ranging from 0 to 128.

- Coefficient 0 is the multiplier for the earliest pixel (P0) in the group of 6 pixels and coefficient 5 is the multiplier for the latest pixel (P5) in the group. The output pixel positional phase is defined as centered in P2 if the positional phase is 0 or proportionally in between P2 and P3 if the positional phase is larger than 0.

Sum of all coefficients for each phase should be 128 typically, and software should never program the sum of all coefficients for a phase to be more than 128.

For each horizontal positional phase, the 6 filter coefficients require 32 register bits. Note that color value ranges from 0 to 255 for Y, R, G, B and -128 to 127 for U and V.

Offset: 0x601 │ Byte Offset: 0x1804 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P00C5: Phase 00 coefficient 5 (typically 0)
28:24	X	C_H_FILTER_P00C4: Phase 00 coefficient 4 (typically 0)
23:16	X	C_H_FILTER_P00C3: Phase 00 coefficient 3 (typically 0)
15:8	X	C_H_FILTER_P00C2: Phase 00 coefficient 2 (typically 128)
7:3	X	C_H_FILTER_P00C1: Phase 00 coefficient 1 (typically 0)
2:0	X	C_H_FILTER_P00C0: Phase 00 coefficient 0 (typically 0)

21.13.4 DC_WINC_C_H_FILTER_P01_0

Window C Horizontal Filter phase 01

Offset: 0x602 │ Byte Offset: 0x1808 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P01C5: Phase 01 coefficient 5 (typically 1)
28:24	X	C_H_FILTER_P01C4: Phase 01 coefficient 4 (typically -2)
23:16	X	C_H_FILTER_P01C3: Phase 01 coefficient 3 (typically 8)
15:8	X	C_H_FILTER_P01C2: Phase 01 coefficient 2 (typically 124)
7:3	X	C_H_FILTER_P01C1: Phase 01 coefficient 1 (typically -4)
2:0	X	C_H_FILTER_P01C0: Phase 01 coefficient 0 (typically 1)

21.13.5 DC_WINC_C_H_FILTER_P02_0

Window C Horizontal Filter phase 02

Offset: 0x603 │ Byte Offset: 0x180c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P02C5: Phase 02 coefficient 5 (typically 1)
28:24	X	C_H_FILTER_P02C4: Phase 02 coefficient 4 (typically -5)
23:16	X	C_H_FILTER_P02C3: Phase 02 coefficient 3 (typically 17)

Bit	Reset	Description
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)

21.13.6 DC_WINC_C_H_FILTER_P03_0

Window C Horizontal Filter phase 03

Offset: 0x604 | Byte Offset: 0x1810 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P03C5: Phase 03 coefficient 5 (typically 2)
28:24	X	C_H_FILTER_P03C4: Phase 03 coefficient 4 (typically -7)
23:16	X	C_H_FILTER_P03C3: Phase 03 coefficient 3 (typically 27)
15:8	X	C_H_FILTER_P03C2: Phase 03 coefficient 2 (typically 115)
7:3	X	C_H_FILTER_P03C1: Phase 03 coefficient 1 (typically -11)
2:0	X	C_H_FILTER_P03C0: Phase 03 coefficient 0 (typically 2)

21.13.7 DC_WINC_C_H_FILTER_P04_0

Window C Horizontal Filter phase 04

Offset: 0x605 | Byte Offset: 0x1814 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P04C5: Phase 04 coefficient 5 (typically 2)
28:24	X	C_H_FILTER_P04C4: Phase 04 coefficient 4 (typically -9)
23:16	X	C_H_FILTER_P04C3: Phase 04 coefficient 3 (typically 37)
15:8	X	C_H_FILTER_P04C2: Phase 04 coefficient 2 (typically 109)
7:3	X	C_H_FILTER_P04C1: Phase 04 coefficient 1 (typically -13)
2:0	X	C_H_FILTER_P04C0: Phase 04 coefficient 0 (typically 2)

21.13.8 DC_WINC_C_H_FILTER_P05_0

Window C Horizontal Filter phase 05

Offset: 0x606 | Byte Offset: 0x1818 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P05C5: Phase 05 coefficient 5 (typically 2)
28:24	X	C_H_FILTER_P05C4: Phase 05 coefficient 4 (typically -11)

Bit	Reset	Description
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)

21.13.9 DC_WINC_C_H_FILTER_P06_0

Window C Horizontal Filter phase 06

Offset: 0x607 | Byte Offset: 0x181c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P06C5: Phase 06 coefficient 5 (typically 3)
28:24	X	C_H_FILTER_P06C4: Phase 06 coefficient 4 (typically -13)
23:16	X	C_H_FILTER_P06C3: Phase 06 coefficient 3 (typically 56)
15:8	X	C_H_FILTER_P06C2: Phase 06 coefficient 2 (typically 94)
7:3	X	C_H_FILTER_P06C1: Phase 06 coefficient 1 (typically -15)
2:0	X	C_H_FILTER_P06C0: Phase 06 coefficient 0 (typically 3)

21.13.10 DC_WINC_C_H_FILTER_P07_0

Window C Horizontal Filter phase 07

Offset: 0x608 | Byte Offset: 0x1820 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P07C5: Phase 07 coefficient 5 (typically 3)
28:24	X	C_H_FILTER_P07C4: Phase 07 coefficient 4 (typically -14)
23:16	X	C_H_FILTER_P07C3: Phase 07 coefficient 3 (typically 67)
15:8	X	C_H_FILTER_P07C2: Phase 07 coefficient 2 (typically 85)
7:3	X	C_H_FILTER_P07C1: Phase 07 coefficient 1 (typically -16)
2:0	X	C_H_FILTER_P07C0: Phase 07 coefficient 0 (typically 3)

21.13.11 DC_WINC_C_H_FILTER_P08_0

Window C Horizontal Filter phase 08

Offset: 0x609 | Byte Offset: 0x1824 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P08C5: Phase 08 coefficient 5 (typically 3)

Bit	Reset	Description
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)

21.13.12 DC_WINC_C_H_FILTER_P09_0

Window C Horizontal Filter phase 09

Offset: 0x60a │ Byte Offset: 0x1828 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P09C5: Phase 09 coefficient 5 (typically 3)
28:24	X	C_H_FILTER_P09C4: Phase 09 coefficient 4 (typically -16)
23:16	X	C_H_FILTER_P09C3: Phase 09 coefficient 3 (typically 85)
15:8	X	C_H_FILTER_P09C2: Phase 09 coefficient 2 (typically 67)
7:3	X	C_H_FILTER_P09C1: Phase 09 coefficient 1 (typically -14)
2:0	X	C_H_FILTER_P09C0: Phase 09 coefficient 0 (typically 3)

21.13.13 DC_WINC_C_H_FILTER_P0A_0

Window C Horizontal Filter phase 0A

Offset: 0x60b │ Byte Offset: 0x182c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P0AC5: Phase 0A coefficient 5 (typically 3)
28:24	X	C_H_FILTER_P0AC4: Phase 0A coefficient 4 (typically -15)
23:16	X	C_H_FILTER_P0AC3: Phase 0A coefficient 3 (typically 94)
15:8	X	C_H_FILTER_P0AC2: Phase 0A coefficient 2 (typically 56)
7:3	X	C_H_FILTER_P0AC1: Phase 0A coefficient 1 (typically -13)
2:0	X	C_H_FILTER_P0AC0: Phase 0A coefficient 0 (typically 3)

21.13.14 DC_WINC_C_H_FILTER_P0B_0

Window C Horizontal Filter phase 0B

Offset: 0x60c │ Byte Offset: 0x1830 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P0BC5: Phase 0B coefficient 5 (typically 3)
28:24	X	C_H_FILTER_P0BC4: Phase 0B coefficient 4 (typically -15)
23:16	X	C_H_FILTER_P0BC3: Phase 0B coefficient 3 (typically 102)
15:8	X	C_H_FILTER_P0BC2: Phase 0B coefficient 2 (typically 47)
7:3	X	C_H_FILTER_P0BC1: Phase 0B coefficient 1 (typically -11)
2:0	X	C_H_FILTER_P0BC0: Phase 0B coefficient 0 (typically 2)

21.13.15 DC_WINC_C_H_FILTER_P0C_0

Window C Horizontal Filter phase 0C

Offset: 0x60d │ Byte Offset: 0x1834 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P0CC5: Phase 0C coefficient 5 (typically 2)
28:24	X	C_H_FILTER_P0CC4: Phase 0C coefficient 4 (typically -13)
23:16	X	C_H_FILTER_P0CC3: Phase 0C coefficient 3 (typically 109)
15:8	X	C_H_FILTER_P0CC2: Phase 0C coefficient 2 (typically 37)
7:3	X	C_H_FILTER_P0CC1: Phase 0C coefficient 1 (typically -9)
2:0	X	C_H_FILTER_P0CC0: Phase 0C coefficient 0 (typically 2)

21.13.16 DC_WINC_C_H_FILTER_P0D_0

Window C Horizontal Filter phase 0D

Offset: 0x60e │ Byte Offset: 0x1838 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P0DC5: Phase 0D coefficient 5 (typically 2)
28:24	X	C_H_FILTER_P0DC4: Phase 0D coefficient 4 (typically -11)
23:16	X	C_H_FILTER_P0DC3: Phase 0D coefficient 3 (typically 115)
15:8	X	C_H_FILTER_P0DC2: Phase 0D coefficient 2 (typically 27)
7:3	X	C_H_FILTER_P0DC1: Phase 0D coefficient 1 (typically -7)
2:0	X	C_H_FILTER_P0DC0: Phase 0D coefficient 0 (typically 2)

21.13.17 DC_WINC_C_H_FILTER_P0E_0

Window C Horizontal Filter phase 0E

Offset: 0x60f │ Byte Offset: 0x183c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P0EC5: Phase 0E coefficient 5 (typically 1)
28:24	X	C_H_FILTER_P0EC4: Phase 0E coefficient 4 (typically -8)
23:16	X	C_H_FILTER_P0EC3: Phase 0E coefficient 3 (typically 122)
15:8	X	C_H_FILTER_P0EC2: Phase 0E coefficient 2 (typically 17)
7:3	X	C_H_FILTER_P0EC1: Phase 0E coefficient 1 (typically -5)
2:0	X	C_H_FILTER_P0EC0: Phase 0E coefficient 0 (typically 1)

21.13.18 DC_WINC_C_H_FILTER_P0F_0

Window C Horizontal Filter phase 0F

Offset: 0x610 │ Byte Offset: 0x1840 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P0FC5: Phase 0F coefficient 5 (typically 1)
28:24	X	C_H_FILTER_P0FC4: Phase 0F coefficient 4 (typically -4)
23:16	X	C_H_FILTER_P0FC3: Phase 0F coefficient 3 (typically 124)
15:8	X	C_H_FILTER_P0FC2: Phase 0F coefficient 2 (typically 8)
7:3	X	C_H_FILTER_P0FC1: Phase 0F coefficient 1 (typically -2)
2:0	X	C_H_FILTER_P0FC0: Phase 0F coefficient 0 (typically 1)

21.13.19 DC_WINC_C_CSC_YOF_0

Window C CSC Y Offset

Color Space Conversion coefficients

The CSC can be used for YUV to RGB conversion with brightness and hue/saturation control. The CSC can only be enabled for window C controlled by CSC_ENABLE register bits.

For Y color, the Y offset is applied first and saturation (clipping) is performed immediately after the Y offset is applied.

- R = sat(KYRGB * sat(Y + YOF) + KUR * U + KVR * V)
- G = sat(KYRGB * sat(Y + YOF) + KUG * U + KVG * V)
- B = sat(KYRGB * sat(Y + YOF) + KUB * U + KVB * V)

Saturation and rounding is performed in the range of 0 to 255 for the above equations.

Typical values are:

- YOF = -16.000, KYRGB = 1.1644

- KUR = 0.0000, KVR = 1.5960
- KUG = -0.3918, KVG = -0.8130
- KUB = 2.0172, KVB = 0.0000

KUR and KVB are typically 0.0000 but they may be programmed non-zero for hue rotation.

The CSC can also take RGB input, in which case YOF, KVB, KUG, and KUR should be programmed to 0, and KYRGB will be forced to 0 by the hardware for generating R and B. KYRGB will not be forced to 0 for generating G. KVR, KYRGB, and KUB can be programmed to 1.0 or used as gain control for R, G, B, correspondingly. Note that color value ranges from 0 to 255 for Y, R, G, B, and -128 to 127 for U and V.

Offset: 0x611 | Byte Offset: 0x1844 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_CSC_YOF: Y Offset in s.7.0 format

21.13.20 DC_WINC_C_CSC_KYRGB_0

Window C CSC Y Coefficient (gain) for RGB

Offset: 0x612 | Byte Offset: 0x1848 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	C_CSC_KYRGB: Y Gain for R, G, B colors in 2.8 format

21.13.21 DC_WINC_C_CSC_KUR_0

Window C CSC U coefficient for R

Offset: 0x613 | Byte Offset: 0x184c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	C_CSC_KUR: U coefficients for R in s.2.8 format

21.13.22 DC_WINC_C_CSC_KVR_0

Window C CSC V coefficient for R

Offset: 0x614 | Byte Offset: 0x1850 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	C_CSC_KVR: V coefficients for R in s.2.8 format

21.13.23 DC_WINC_C_CSC_KUG_0

Window C CSC U coefficient for G

Offset: 0x615 | Byte Offset: 0x1854 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	C_CSC_KUG: U coefficients for G in s.1.8 format

21.13.24 DC_WINC_C_CSC_KVG_0

Window C CSC V coefficient for G

Offset: 0x616 | Byte Offset: 0x1858 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	C_CSC_KVG: V coefficients for G in s.1.8 format

21.13.25 DC_WINC_C_CSC_KUB_0

Window C CSC U coefficient for B

Offset: 0x617 | Byte Offset: 0x185c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	C_CSC_KUB: U coefficients for B in s.2.8 format

21.13.26 DC_WINC_C_CSC_KVB_0

Window C CSC V coefficient for B

Offset: 0x618 | Byte Offset: 0x1860 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	C_CSC_KVB: V coefficients for B in s.2.8 format

21.13.27 DC_WINC_C_V_FILTER_P00_0

Window C Vertical Filter phase 00

Vertical scaling filter coefficients

Vertical scaling filter is a 2-tap filter with 4-bit positional phase. Coefficients 0 and 1 are 8-bit unsigned values ranging from 0 to 128.

Coefficient 0 is the multiplier for the earlier pixel (P0) in the group of 2-pixels and coefficient 1 is the multiplier for the later pixel (P1) in the group. The output pixel positional phase is defined as centered in P0 if the positional phase is 0 or proportionally in between P0 and P1 if the positional phase is larger than 0.

Sum of all coefficients for each phase should be 128 typically; therefore coefficient 1 can be calculated from (1 - coefficient 0) and only coefficient 0 is programmed.

For each vertical positional phase, the filter coefficient requires 8 register bits. Note that color value ranges from 0 to 255 for Y, R, G, B and -128 to 127 for U and V.

Offset: 0x619 | Byte Offset: 0x1864 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P00C0: Phase 00 coefficient 0 (typically 128)

21.13.28 DC_WINC_C_V_FILTER_P01_0

Window C Vertical Filter phase 01

Offset: 0x61a | Byte Offset: 0x1868 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P01C0: Phase 01 coefficient 0 (typically 120)

21.13.29 DC_WINC_C_V_FILTER_P02_0

Window C Vertical Filter phase 02

Offset: 0x61b | Byte Offset: 0x186c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P02C0: Phase 02 coefficient 0 (typically 112)

21.13.30 DC_WINC_C_V_FILTER_P03_0

Window C Vertical Filter phase 03

Offset: 0x61c | Byte Offset: 0x1870 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P03C0: Phase 03 coefficient 0 (typically 104)

21.13.31 DC_WINC_C_V_FILTER_P04_0

Window C Vertical Filter phase 04

Offset: 0x61d | Byte Offset: 0x1874 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P04C0: Phase 04 coefficient 0 (typically 96)

21.13.32 DC_WINC_C_V_FILTER_P05_0

Window C Vertical Filter phase 05

Offset: 0x61e | Byte Offset: 0x1878 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P05C0: Phase 05 coefficient 0 (typically 88)

21.13.33 DC_WINC_C_V_FILTER_P06_0

Window C Vertical Filter phase 06

Offset: 0x61f | Byte Offset: 0x187c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P06C0: Phase 06 coefficient 0 (typically 80)

21.13.34 DC_WINC_C_V_FILTER_P07_0

Window C Vertical Filter phase 07

Offset: 0x620 | Byte Offset: 0x1880 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P07C0: Phase 07 coefficient 0 (typically 72)

21.13.35 DC_WINC_C_V_FILTER_P08_0

Window C Vertical Filter phase 08

Offset: 0x621 | Byte Offset: 0x1884 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P08C0: Phase 08 coefficient 0 (typically 64)

21.13.36 DC_WINC_C_V_FILTER_P09_0

Window C Vertical Filter phase 09

Offset: 0x622 | Byte Offset: 0x1888 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P09C0: Phase 09 coefficient 0 (typically 56)

21.13.37 DC_WINC_C_V_FILTER_P0A_0

Window C Vertical Filter phase 0A

Offset: 0x623 | Byte Offset: 0x188c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P0AC0: Phase 0A coefficient 0 (typically 48)

21.13.38 DC_WINC_C_V_FILTER_P0B_0

Window C Vertical Filter phase 0B

Offset: 0x624 | Byte Offset: 0x1890 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P0BC0: Phase 0B coefficient 0 (typically 40)

21.13.39 DC_WINC_C_V_FILTER_P0C_0

Window C Vertical Filter phase 0C

Offset: 0x625 | Byte Offset: 0x1894 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P0CC0: Phase 0C coefficient 0 (typically 32)

21.13.40 DC_WINC_C_V_FILTER_P0D_0

Window C Vertical Filter phase 0D

Offset: 0x626 | Byte Offset: 0x1898 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P0DC0: Phase 0D coefficient 0 (typically 24)

21.13.41 DC_WINC_C_V_FILTER_P0E_0

Window C Vertical Filter phase 0E

Offset: 0x627 | Byte Offset: 0x189c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P0EC0: Phase 0E coefficient 0 (typically 16)

21.13.42 DC_WINC_C_V_FILTER_P0F_0

Window C Vertical Filter phase 0F

Offset: 0x628 | Byte Offset: 0x18a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P0FC0: Phase 0F coefficient 0 (typically 8)

21.13.43 DC_WINC_C_H_FILTER_HI_P00_0

Window C Horizontal Filter phase 00

Offset: 0x629 | Byte Offset: 0x18a4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	C_H_FILTER_HI_P00C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	C_H_FILTER_HI_P00C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	C_H_FILTER_HI_P00C3: MSB 9:9 of the lobe 3.
4	X	C_H_FILTER_HI_P00C2: MSB 9:9 of the lobe 2.
3:2	X	C_H_FILTER_HI_P00C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	C_H_FILTER_HI_P00C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.13.44 DC_WINC_C_H_FILTER_HI_P01_0

Window C Horizontal Filter phase 01

Offset: 0x62a | Byte Offset: 0x18a8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	C_H_FILTER_HI_P01C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information.

Bit	Reset	Description
		LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	C_H_FILTER_HI_P01C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	C_H_FILTER_HI_P01C3: MSB 9:9 of the lobe 3.
4	X	C_H_FILTER_HI_P01C2: MSB 9:9 of the lobe 2.
3:2	X	C_H_FILTER_HI_P01C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	C_H_FILTER_HI_P01C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.13.45 DC_WINC_C_H_FILTER_HI_P02_0

Window C Horizontal Filter phase 02

Offset: 0x62b | Byte Offset: 0x18ac | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	C_H_FILTER_HI_P02C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	C_H_FILTER_HI_P02C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	C_H_FILTER_HI_P02C3: MSB 9:9 of the lobe 3.
4	X	C_H_FILTER_HI_P02C2: MSB 9:9 of the lobe 2.
3:2	X	C_H_FILTER_HI_P02C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	C_H_FILTER_HI_P02C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.13.46 DC_WINC_C_H_FILTER_HI_P03_0

Window C Horizontal Filter phase 03

Offset: 0x62c | Byte Offset: 0x18b0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	C_H_FILTER_HI_P03C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	C_H_FILTER_HI_P03C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	C_H_FILTER_HI_P03C3: MSB 9:9 of the lobe 3.
4	X	C_H_FILTER_HI_P03C2: MSB 9:9 of the lobe 2.
3:2	X	C_H_FILTER_HI_P03C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	C_H_FILTER_HI_P03C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.13.47 DC_WINC_C_H_FILTER_HI_P04_0

Window C Horizontal Filter phase 04

Offset: 0x62d | Byte Offset: 0x18b4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	C_H_FILTER_HI_P04C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	C_H_FILTER_HI_P04C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	C_H_FILTER_HI_P04C3: MSB 9:9 of the lobe 3.
4	X	C_H_FILTER_HI_P04C2: MSB 9:9 of the lobe 2.
3:2	X	C_H_FILTER_HI_P04C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	C_H_FILTER_HI_P04C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.13.48 DC_WINC_C_H_FILTER_HI_P05_0

Window C Horizontal Filter phase 05

Offset: 0x62e | Byte Offset: 0x18b8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	C_H_FILTER_HI_P05C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	C_H_FILTER_HI_P05C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	C_H_FILTER_HI_P05C3: MSB 9:9 of the lobe 3.
4	X	C_H_FILTER_HI_P05C2: MSB 9:9 of the lobe 2.
3:2	X	C_H_FILTER_HI_P05C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	C_H_FILTER_HI_P05C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.13.49 DC_WINC_C_H_FILTER_HI_P06_0

Window C Horizontal Filter phase 06

Offset: 0x62f | Byte Offset: 0x18bc | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	C_H_FILTER_HI_P06C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	C_H_FILTER_HI_P06C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	C_H_FILTER_HI_P06C3: MSB 9:9 of the lobe 3.
4	X	C_H_FILTER_HI_P06C2: MSB 9:9 of the lobe 2.
3:2	X	C_H_FILTER_HI_P06C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.

Bit	Reset	Description
1:0	X	C_H_FILTER_HI_P06C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.13.50 DC_WINC_C_H_FILTER_HI_P07_0

Window C Horizontal Filter phase 07

Offset: 0x630 | Byte Offset: 0x18c0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	C_H_FILTER_HI_P07C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	C_H_FILTER_HI_P07C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	C_H_FILTER_HI_P07C3: MSB 9:9 of the lobe 3.
4	X	C_H_FILTER_HI_P07C2: MSB 9:9 of the lobe 2.
3:2	X	C_H_FILTER_HI_P07C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	C_H_FILTER_HI_P07C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.13.51 DC_WINC_C_H_FILTER_HI_P08_0

Window C Horizontal Filter phase 08

Offset: 0x631 | Byte Offset: 0x18c4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	C_H_FILTER_HI_P08C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	C_H_FILTER_HI_P08C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	C_H_FILTER_HI_P08C3: MSB 9:9 of the lobe 3.
4	X	C_H_FILTER_HI_P08C2: MSB 9:9 of the lobe 2.
3:2	X	C_H_FILTER_HI_P08C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	C_H_FILTER_HI_P08C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.13.52 DC_WINC_C_H_FILTER_HI_P09_0

Window C Horizontal Filter phase 09

Offset: 0x632 | Byte Offset: 0x18c8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	C_H_FILTER_HI_P09C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	C_H_FILTER_HI_P09C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.

Bit	Reset	Description
5	X	C_H_FILTER_HI_P09C3: MSB 9:9 of the lobe 3.
4	X	C_H_FILTER_HI_P09C2: MSB 9:9 of the lobe 2.
3:2	X	C_H_FILTER_HI_P09C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	C_H_FILTER_HI_P09C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.13.53 DC_WINC_C_H_FILTER_HI_P0A_0

Window C Horizontal Filter phase 0A

Offset: 0x633 | Byte Offset: 0x18cc | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	C_H_FILTER_HI_P0AC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	C_H_FILTER_HI_P0AC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	C_H_FILTER_HI_P0AC3: MSB 9:9 of the lobe 3.
4	X	C_H_FILTER_HI_P0AC2: MSB 9:9 of the lobe 2.
3:2	X	C_H_FILTER_HI_P0AC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	C_H_FILTER_HI_P0AC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.13.54 DC_WINC_C_H_FILTER_HI_P0B_0

Window C Horizontal Filter phase 0B

Offset: 0x634 | Byte Offset: 0x18d0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	C_H_FILTER_HI_P0BC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	C_H_FILTER_HI_P0BC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	C_H_FILTER_HI_P0BC3: MSB 9:9 of the lobe 3.
4	X	C_H_FILTER_HI_P0BC2: MSB 9:9 of the lobe 2.
3:2	X	C_H_FILTER_HI_P0BC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	C_H_FILTER_HI_P0BC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.13.55 DC_WINC_C_H_FILTER_HI_P0C_0

Window C Horizontal Filter phase 0C

Offset: 0x635 | Byte Offset: 0x18d4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	C_H_FILTER_HI_P0CC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	C_H_FILTER_HI_P0CC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	C_H_FILTER_HI_P0CC3: MSB 9:9 of the lobe 3.
4	X	C_H_FILTER_HI_P0CC2: MSB 9:9 of the lobe 2.
3:2	X	C_H_FILTER_HI_P0CC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	C_H_FILTER_HI_P0CC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.13.56 DC_WINC_C_H_FILTER_HI_P0D_0

Window C Horizontal Filter phase 0D

Offset: 0x636 | Byte Offset: 0x18d8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	C_H_FILTER_HI_P0DC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	C_H_FILTER_HI_P0DC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	C_H_FILTER_HI_P0DC3: MSB 9:9 of the lobe 3.
4	X	C_H_FILTER_HI_P0DC2: MSB 9:9 of the lobe 2.
3:2	X	C_H_FILTER_HI_P0DC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	C_H_FILTER_HI_P0DC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.13.57 DC_WINC_C_H_FILTER_HI_P0E_0

Window C Horizontal Filter phase 0E

Offset: 0x637 | Byte Offset: 0x18dc | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	C_H_FILTER_HI_P0EC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	C_H_FILTER_HI_P0EC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	C_H_FILTER_HI_P0EC3: MSB 9:9 of the lobe 3.
4	X	C_H_FILTER_HI_P0EC2: MSB 9:9 of the lobe 2.
3:2	X	C_H_FILTER_HI_P0EC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.

Bit	Reset	Description
1:0	X	C_H_FILTER_HI_P0EC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

21.13.58 DC_WINC_C_H_FILTER_HI_P0F_0

Window C Horizontal Filter phase 0F

Offset: 0x638 | Byte Offset: 0x18e0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	C_H_FILTER_HI_P0FC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	C_H_FILTER_HI_P0FC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	C_H_FILTER_HI_P0FC3: MSB 9:9 of the lobe 3.
4	X	C_H_FILTER_HI_P0FC2: MSB 9:9 of the lobe 2.
3:2	X	C_H_FILTER_HI_P0FC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	C_H_FILTER_HI_P0FC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

The registers under DC_WIN are double buffered.

21.13.59 DC_WIN_C_WIN_OPTIONS_0

Window C Options

Class: Display Window Settings

Display Window C parameters

Offset: 0x700 │ Byte Offset: 0x1c00 │ Read/Write: R/W │ Reset: 0x00XXXXXX (0b00xxxxxx0xxxxxx0x0xxxxxxxxx0xxxx)

Bit	Reset	Description
31	0x0	c_H_FILTER_MODE: Horizontal filter mode. 0 = With previous Tegra horizontal coeffients widths. 0,5 - signed 5 bits 1,4 - signed 9 bits 2,3 - unsigned 9 bits 1 = New mode. With the expanded horizontal filter coefficients widths. 0,5- signed 5 bits 1,4- signed 7 bits 2,3- unsigned 9 bits 0 = OLD 1 = NEW
30	0x0	C_WIN_ENABLE: Window C Window enable 0 = DISABLE 1 = ENABLE
23	0x0	C_INTERLACE_ENABLE: Window C Interlace enable. This controls the fetch unit toggle between odd and even (normal) base addresses at the start of each field. Base address select between odd/even fields is determined by HVTGEN field status. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
22	X	C_YUV_RANGE_EXPAND: Window C Enable range expansion in the cases where RANGEREDFRM is 1 from mpd. Formula: Y = clip((Y-128)*2 + 128); Cb = clip((Cb-128)*2 + 128); Cr = clip((Cr-128)*2 + 128); where clip() function clips between 0 and 255. 0 = DISABLE 1 = ENABLE
20	X	C_DV_ENABLE: Window C Digital Vibrance Enable 0 = DISABLE 1 = ENABLE
18	X	C_CSC_ENABLE: Window C Color Space Conversion Enable. This controls the color space conversion and should be enabled for YCbCr/YUV color modes for conversion to B8G8R8 and for hue and saturation control. This can also be used for gain control for RGB color modes. 0 = DISABLE 1 = ENABLE
16	0x0	C_CP_ENABLE: Window C Color Palette Enable. This controls the color palette and should be enabled for palletized color modes. For non-palletized color modes, the color palette can be enabled for gamma correction. 0 = DISABLE 1 = ENABLE
14	0x0	C_V_FILTER_UV_ALIGN: Window C V Filter UV Alignment. This is effective only when vertical scaling filter is enabled and only on these formats YCbCr420P, YUV420P, YCbCr422R, YUV422R, YCbCr422RA YUV422RA. When UV alignment is enabled, the chroma components are aligned to the even number of luma component lines. When disabled the chroma components are aligned to half a pixel below the corresponding even number of luma component lines. It is usually disabled unless the incoming video stream specifically indicates otherwise. 0 = DISABLE 1 = ENABLE
12	X	C_V_FILTER_OPTIMIZE: Window C V Filter Optimization. This is effective only when vertical scaling filter is enabled. This can be used (enabled) to temporarily disable the vertical scaling filter when the vertical scaling DDA fraction is zero. In this case the next line is not fetched from memory to save bandwidth and power. This feature cannot be used in 420P/422R/422RA formats. 0 = DISABLE 1 = ENABLE
10	X	C_V_FILTER_ENABLE: Window C V Filter Enable. This controls the V scaling filter and is effective only for non-palletized color modes. If V filter is disabled, only one line is read from memory for each output line. NOTE: This feature is not supported. This bit is ignored and is a don't care. 0 = DISABLE 1 = ENABLE
8	X	C_H_FILTER_ENABLE: Window C H Filter Enable. This controls H scaling filter and is effective only for non-palletized color modes. 0 = DISABLE 1 = ENABLE
6	X	C_COLOR_EXPAND: Window C 12/15/16/18-to-24 bpp color expansion. This bit should be enabled only for 12-bpp B4G4R4A4, 15-bpp B5G5R5A, 16-bpp B5G6R5, 18-bpp B6G6R6 color modes. If enabled, the color conversion is performed prior to horizontal scaling. 0 = DISABLE 1 = ENABLE
4	0x0	C_SCAN_COLUMN: Window C Scanning direction. 0= Scan in horizontal direction (for 0 or 180 degrees). 1= Scan in vertical direction (for 90 or 270 degrees) 0 = DISABLE 1 = ENABLE

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

21.13.60 DC_WIN_C_BYTE_SWAP_0

Window C Byte Swap

Offset: 0x701 │ Byte Offset: 0x1c04 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2:0	X	C_BYTE_SWAP: Window C Byte Swap. This controls byte swap of frame data read from memory prior to any data processing in the display module. 000= no byte swap (3 2 1 0) 001= byte swap for each 2-byte word (2 3 0 1) 010= byte swap for each 4-byte word (0 1 2 3) 011= word swap for each 4-byte word (1 0 3 2) 100= byte0 swapped with byte2 (3 0 1 2) 101= left shift every byte (2 1 0 3) 0 = NOSWAP 1 = SWAP2 2 = SWAP4 3 = SWAP4HW 4 = SWAP02 5 = SWAPLEFT

21.13.61 DC_WIN_C_COLOR_DEPTH_0

Window C Color Depth. For YCbCr data format, Cb and Cr are 8-bit unsigned values. For YUV data format, U and V are 8-bit signed values. YCbCr422R is similar to YCbCr422P but the Cb and Cr are shared vertically.

YUV422R is similar to YUV422P, but the U and V are shared vertically. YCbCr422RA is the same as YCbCr422R in memory and YUV422RA is the same as YUV422R in memory but while reading from memory, for YCbCr422RA and YUV422RA, chroma averaging is applied for each pixel pair so that they can be processed as YUV422 by the display pipeline.

For YCbCr422R and YUV422R, every other chroma pixels are not used (discarded) by the display pipeline. B6x2G6x2R6x2A8 is similar to B8G8R6A8 but with the 2 LSBs zeroed out. R6x2G6x2B6x2A8 is similar to R8G8B6A8 but with the 2 LSBs zeroed out.

Some formats have NVCF_ or T_ prefix aliases for NVidia surface format naming convention, wherein left-to-right component names correspond to MSB to LSB bits within a word. In old names without NVCF_ or T_, left-to-right component names generally map LSB to MSB within a word.

Offset: 0x703 │ Byte Offset: 0x1c0c │ Read/Write: R/W │ Reset: 0x0000000c (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0001100))

Bit	Reset	Description
6:0	B8G8R8A8	C_COLOR_DEPTH: Window C Color Depth Supported color depths are: P8 = 8-bpp (palletized) B4G4R4A4 = 12-bpp B4G4R4 B5G5R5A = 15-bpp B5G5R5 AB5G5R5 = 15-bpp B5G5R5 B5G6R5 = 16-bpp B5G6R5 B8G8R8A8 = 32-bpp B8G8R8A8 R8G8B8A8 = 32-bpp R8G8B8A8 YCbCr422 = 16-bpp YCbCr422 packed YUV422 = 16-bpp YUV422 YCbCr420P = 16-bpp YCbCr420 planar YUV420P = 16-bpp YUV420 planar YCbCr422P = 16-bpp YCbCr422 planar YUV422P = 16-bpp YUV422 planar YCbCr422R = 16-bpp YCbCr422 rotated planar YUV422R = 16-bpp YUV422 rotated planar YCbCr422RA= 16-

Bit	Reset	Description
		bpp YCbCr422 rotated planar with chroma averaging YUV422RA = 16-bpp YUV422 rotated planar with chroma averaging. Reserved values: 0,1,2,14,15,24,25. 3 = T_P8, P8 4 = T_A4R4G4B4, B4G4R4A4 5 = T_A1R5G5B5, B5G5R5A 6 = T_R5G6B5, B5G6R5 7 = T_R5G5B5A1, AB5G5R5 12 = T_A8R8G8B8, B8G8R8A8 13 = T_A8B8G8R8, R8G8B8A8 16 = T_U8_Y8__V8_Y8, CbYCrY422, YCbCr422 17 = T_U8_Y8__V8_Y8_TRUE, UYVY422, YUV422 18 = T_Y8___U8___V8_N420, YCbCr420P 19 = T_Y8___U8___V8_N420_TRUE, YUV420P 20 = T_Y8___U8___V8_N422, YCbCr422P 21 = T_Y8___U8___V8_N422_TRUE, YUV422P 22 = T_Y8___U8___V8_N422R, YCbCr422RP, YCbCr422R 23 = T_Y8___U8___V8_N422R_TRUE, YUV422RP, YUV422R 24 = T_V8_Y8__U8_Y8, CrYCbY422 25 = T_V8_Y8__U8_Y8_TRUE, VYUY422 26 = T_Y8, Y8 27 = T_A4B4G4R4, R4G4B4A4 28 = T_A1B5G5R5, R5G5B5A 29 = T_B5G5R5A1, AR5G5B5 30 = T_X1R5G5B5, B5G5R5X1 31 = T_R5G5B5X1, X1B5G5R5 32 = T_X1B5G5R5, R5G5B5X1 33 = T_B5G5R5X1, X1R5G5B5 34 = T_B5G6R5, R5G6B5 35 = T_B8G8R8A8 36 = T_R8G8B8A8 37 = T_X8R8G8B8, B8G8R8X8 38 = T_X8B8G8R8, R8G8B8X8 39 = T_A8Y8U8V8 40 = T_V8U8Y8A8 41 = T_Y8___U8___V8_N444, YCbCr444P 42 = T_Y8___U8V8_N420, YCrCb420SP 43 = T_Y8___V8U8_N420, YCbCr420SP 44 = T_Y8___U8V8_N422, YCrCb422SP 45 = T_Y8___V8U8_N422, YCbCr422SP 46 = T_Y8___U8V8_N422R, YCrCb422RSP 47 = T_Y8___V8U8_N422R, YCbCr422RSP 48 = T_Y8___U8V8_N444, YCrCb444SP 49 = T_Y8___V8U8_N444, YCbCr444SP 50 = T_A8Y8U8V8_TRUE 51 = T_V8U8Y8A8_TRUE 52 = T_Y8___U8___V8_N444_TRUE, YUV444P 53 = T_Y8___U8V8_N420_TRUE, YVU420SP 54 = T_Y8___V8U8_N420_TRUE, YUV420SP 55 = T_Y8___U8V8_N422_TRUE, YVU422SP 56 = T_Y8___V8U8_N422_TRUE, YUV422SP 57 = T_Y8___U8V8_N422R_TRUE, YVU422RSP 58 = T_Y8___V8U8_N422R_TRUE, YUV422RSP 59 = T_Y8___U8V8_N444_TRUE, YVU444SP 60 = T_Y8___V8U8_N444_TRUE, YUV444SP 61 = T_Y8_U8__Y8_V8, YCbYCr422 62 = T_Y8_U8__Y8_V8_TRUE, YUYV422 63 = T_Y8_V8__Y8_U8, YCrYCb422 64 = T_Y8_V8__Y8_U8_TRUE, YVYU422 65 = T_R8G8B8X8 66 = T_B8G8R8X8: new formats, reserved for future use

21.13.62 DC_WIN_C_POSITION_0

Window C Position

This register defines H position and size of Window C after scaling (if there is any)

Offset: 0x704 │ Byte Offset: 0x1c10 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	C_V_POSITION: Window C V Position. This is specified with respect to the top edge of active display area.
12:0	X	C_H_POSITION: Window C H Position. This is specified with respect to the left edge of active display area.

21.13.63 DC_WIN_C_SIZE_0

Window C Size

This register defines V position and size of Window C after scaling (if there is any)

Note: Programming on window size should guarantee the whole window is inside the active area, otherwise extra rows/columns will be fetched and discarded which affects performance.

Offset: 0x705 │ Byte Offset: 0x1c14 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	C_V_SIZE: Window C V Size (lines). This is the vertical size after scaling.
12:0	X	C_H_SIZE: Window C H Size (pixels). This is the horizontal size after scaling.

21.13.64 DC_WIN_C_PRESCALED_SIZE_0

Window C Pre-scaled Size

This register defines Window C pre-scaled size.

The H pre-scaled size is needed to determine how many bytes to fetch from memory per line and this parameter must be programmed exactly as needed taking into account the scaling factor. For planar YUV or YCbCr data formats, this parameter refer to the H pre-scaled of the Y plane.

The total number of lines to be fetched from memory is determined by post-scale V size but V pre-scaled size is needed to 'clamp' the last valid line if the vertical DDA is exactly or slightly beyond the specified V pre-scaled size.

H pre-scaled size ideally should be in terms of pixel but then hardware needs to convert this precisely to bytes to determine the amount of data to request from memory.

Offset: 0x706 │ Byte Offset: 0x1c18 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	C_V_PRESCALED_SIZE: Window C V Pre-scaled Size (lines). In 420P/422R/422RA formats, it must be even.
14:0	X	C_H_PRESCALED_SIZE: Window C H Pre-scaled Size (bytes). In 420P and 422P formats, it must be even.

21.13.65 DC_WIN_C_H_INITIAL_DDA_0

Window C H Initial DDA

The first pixel of the pre-scaled image is always used to output the first pixel, so essentially this is the same as forcing the H Initial DDA integer portion to 1 initially even though user typically programs this to 0. If it makes the implementation easier, it is possible to force software to program the Initial DDA integer portion to 1. Similarly with the V Initial DDA.

Offset: 0x707 │ Byte Offset: 0x1c1c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	C_H_INITIAL_DDA: Window C H Initial DDA (4.12). This is typically programmed to 0.0

21.13.66 DC_WIN_C_V_INITIAL_DDA_0

Window C V Initial DDA

Offset: 0x708 │ Byte Offset: 0x1c20 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	C_V_INITIAL_DDA: Window C V Initial DDA (4.12). This is typically programmed to 0.0 for both non-interlaced and interlaced sources.

21.13.67 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)) + initial_dda_bias, MAX)
- Filter off: min(round(prescaled_size_in_pixels * 0x1000 / (post_scaled_size_in_pixels - 1) - 0.5) + initial_dda_bias, 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/pixel formats.

  8.0 (0x8000) for 2 Bytes/pixel formats.

Where the value of initial_dda_bias is as follows:

For V_DDA_INCREMENT: initial_dda_bias = (v_intial_dda == negative) ? mod(v_initial_dda)/(post_scaled_size_in_pixels - 1) : 0 ;

For H_DDA_INCREMENT: initial_dda_bias = 0;

They are theoretically the biggest values that guarantee not displaying beyond an image boundary. If the DDA increment is less than 1.0 then image will be upscaled and if DDA increment is more than 1.0, then the image will be downscaled.

Offset: 0x709 | Byte Offset: 0x1c24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	C_V_DDA_INCREMENT: Window C Vertical DDA Increment (4.12). This should be set to 1.0 if there is no scaling. Maximum value is 15.0 regardless of the number of bytes per pixel.
15:0	X	C_H_DDA_INCREMENT: Window C Horizontal DDA Increment (4.12). This should be set to 1.0 if there is no scaling. The maximum value for downscaling depends on the number of bytes per pixel. For 4-byte/pixel modes (32-bpp) the maximum value is 4.0 and for all other modes the maximum value is 8.0.

21.13.68 DC_WIN_C_LINE_STRIDE_0

Window C Line Stride

Offset: 0x70a | Byte Offset: 0x1c28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	C_UV_LINE_STRIDE: Window C Line Stride for Chroma. This is stride (in bytes) for planar YUV or YCbCr data formats for the chroma plane, with the restriction that it must be programmed to be multiples of 4 (16 if tiled or in horizontal flipping) This is not used (ignored) for other non-planar data formats.
15:0	X	C_LINE_STRIDE: Window C Line Stride. This is stride (in bytes) for all non-planar data formats. If the memory surface is tiled, the stride needs to be a multiple of 16. If H_DIRECTION of window C is set to DECREMENT, the stride also needs to be a multiple of 16. For planar YUV or YCbCr data formats, this is stride (in bytes) for the luma plane with the restriction that it must be multiples of 8 (16 if tiled or in horizontal flipping) For tiled surface this value may affect starting address of a window. Refer to the comment of START_ADDR for detail.

21.13.69 DC_WIN_C_DV_CONTROL_0

Window C Digital Vibrance Control

If enabled, Digital Vibrance is applied after H and V scaling and after color palette or color space conversion logic but before color keying multiplexer and before cursor multiplexer.

- After DV, new R = R + (2R - G - B) * FR, where FR is fraction from 0 to 7/8
- After DV, new G = G + (2G - R - B) * FG, where FG is fraction from 0 to 7/8
- After DV, new B = B + (2B - R - G) * FB, where FB is fraction from 0 to 7/8

Offset: 0x70e | Byte Offset: 0x1c38 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18:16	X	C_DV_CONTROL_B: Digital Vibrance control for B
10:8	X	C_DV_CONTROL_G: Digital Vibrance control for G
2:0	X	C_DV_CONTROL_R: Digital Vibrance control for R

Class: Display Color Keying and Blending

Color keying and blending of the display windows are done prior to cursor blending. Cursor always goes on top of the blended windows. Blending is controlled independently on each possible overlap area of the display windows. If 3 windows are enabled there are 7 possible overlap area combinations. For every window in each overlap area combination, color key can be

disabled or enabled. Also, for every window in each overlap area combination, there is a corresponding window blend control parameter and a window blend weight parameter. The window blend control parameter is always effective but the window blend weight is not always used. The window blend weight can also be derived from pixel alpha value or from the reverse of other overlapping windows weight.

Color keying has the highest priority for display window blending. Color key consists of a range of color which is searched independently for each window. If more than 1 windows color key is enabled, then Window A color key has the highest priority, followed by Window B color key, and then followed by Window C color key. Two sets of color key ranges (Color Key 0 and Color Key 1) can be defined; they are shared for all windows. It is possible to use both color key sets for the same window or for two separate windows.

The two sets of color key ranges should not overlap. If they do, the overlapped colors are treated as if they are part of Color Key 0 and not part of Color Key 1.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is enabled for any overlap condition and the window pixel is not within the color key range (key not match), then the window pixel will not be blended with other overlapping window pixels but it will be weighted. The weight is not dependent on the overlap condition it is controlled by the same set of parameters for all overlapping condition.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is enabled for a particular overlap condition and the window pixel is within the color key range (key match), then the window pixel will be weighted and blended with other overlapping pixels and this is controlled separately for each overlap condition.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is disabled then the window pixel will be blended with other overlapping pixels and this is controlled separately for each overlap condition.

Display Color Key Parameters

For B4G4R4A4, B5G6R5A, B5G6R5 modes, the color key should be compared prior to the color conversion to 24-bpp and the unused least significant bits of the pixel are filled with zeros.

For palletized mode, the color key is compared prior to the color palette, and the palletized color is compared against the green color key values/mask.

For YUV mode, U and V are offset by +128 before performing the color key comparison.

In all cases, the color key is compared prior to the horizontal/vertical scaling filter and prior to digital vibrance control.

Both upper and lower values are inclusive.

21.13.70 DC_WIN_C_BLEND_LAYER_CONTROL_0

Window C

Offset: 0x716 | Byte Offset: 0x1c58 | Read/Write: R/W | Reset: 0x01000000 (0bxxxx00010000000000000000000000)

Bit	Reset	Description
27:25	0x0	C_COLOR_KEY_SELECT: 0 = NONE 1 = WINDOWA_KEY0 2 = WINDOWA_KEY1 3 = WINDOWB_KEY0 4 = WINDOWB_KEY1 5 = WINDOWC_KEY0 6 = WINDOWC_KEY1
24	BLEND_BYPASS	C_BLEND_BYPASS: 0 = BLEND_ENABLE 1 = BLEND_BYPASS

Bit	Reset	Description
23:16	0x0	C_K2
15:8	0x0	C_K1
7:0	0x0	C_WINDOW_LAYER_DEPTH

21.13.71 DC_WIN_C_BLEND_MATCH_SELECT_0

Offset: 0x717 | Byte Offset: 0x1c5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xx00x000x000)

Bit	Reset	Description
13:12	0x0	C_BLEND_FACTOR_DST_ALPHA_MATCH_SELECT: 0 = ZERO 1 = ONE 2 = NEG_K1_TIMES_SRC 3 = K2
9:8	0x0	C_BLEND_FACTOR_SRC_ALPHA_MATCH_SELECT: 0 = ZERO 1 = K1 2 = K2
6:4	0x0	C_BLEND_FACTOR_DST_COLOR_MATCH_SELECT: 0 = ZERO 1 = ONE 2 = K1 3 = K2 4 = K1_TIMES_DST 5 = NEG_K1_TIMES_DST 6 = NEG_K1_TIMES_SRC 7 = NEG_K1
2:0	0x0	C_BLEND_FACTOR_SRC_COLOR_MATCH_SELECT: 0 = ZERO 1 = ONE 2 = K1 3 = K1_TIMES_DST 4 = NEG_K1_TIMES_DST 5 = K1_TIMES_SRC

21.13.72 DC_WIN_C_BLEND_NOMATCH_SELECT_0

Offset: 0x718 | Byte Offset: 0x1c60 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xx00x000x000)

Bit	Reset	Description
13:12	0x0	C_BLEND_FACTOR_DST_ALPHA_NOMATCH_SELECT: 0 = ZERO 1 = ONE 2 = NEG_K1_TIMES_SRC 3 = K2
9:8	0x0	C_BLEND_FACTOR_SRC_ALPHA_NOMATCH_SELECT: 0 = ZERO 1 = K1 2 = K2
6:4	0x0	C_BLEND_FACTOR_DST_COLOR_NOMATCH_SELECT: 0 = ZERO 1 = ONE 2 = K1 3 = K2 4 = K1_TIMES_DST 5 = NEG_K1_TIMES_DST 6 = NEG_K1_TIMES_SRC 7 = NEG_K1

Bit	Reset	Description
2:0	0x0	C_BLEND_FACTOR_SRC_COLOR_NOMATCH_SELECT: 0 = ZERO 1 = ONE 2 = K1 3 = K1_TIMES_DST 4 = NEG_K1_TIMES_DST 5 = K1_TIMES_SRC

21.13.73 DC_WIN_C_BLEND_ALPHA_1BIT_0

Offset: 0x719 | Byte Offset: 0x1c64 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	C_BLEND_WEIGHT1: This is used only for 1-bit alpha and alpha value of 1.
7:0	0x0	C_BLEND_WEIGHT0: This is used only for 1-bit alpha and alpha value of 0.

21.14 WINBUF_C Registers

The registers under DC_WINBUF are triple-buffered.

21.14.1 DC_WINBUF_C_START_ADDR_0

Window C Start Address

Overview

START_ADDR, BUF_STRIDE, LINE_STRIDE, ADDR_H_OFFSET, ADDR_V_OFFSET combined, specify the starting address of a window buffer in the memory surface. Generally START_ADDR is programmed with the starting address of the memory surface. H/V_OFFSET specifies the address offsets of the pixel at the beginning of the window, with respect to the starting address of the memory surface. (There is an exception to that when memory surface is not well aligned, see “For linear address mode” under Programming Restrictions below.)

Note that "beginning of the window" here means the top-left corner of a window in normal mode, top-right corner in horizontal flipping, bottom-left corner in vertical flipping, and bottom-right in horizontal+vertical flipping.

Starting Address Calculation

These formulae are same for both tiled and linear modes. However, for linear mode, the addresses calculated are real physical addresses. For tiled mode, these addresses will be translated to the real physical addresses by the memory controller client before being used.

- When a window is host triggered, the starting address of a window is calculated as follows by hardware.
  - non-YUV-planar modes

    starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - YUV-planar modes

    Y-starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    U-starting-address = START_ADDR_U + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    V-starting-address = START_ADDR_V + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by hardware.

- When a window is non-host triggered, starting address of a window buffer is calculated as below.
  - non-YUV-planar mode

    starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - YUV-planar mode:

    Y-starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    U-starting-address = START_ADDR_U + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    V-starting-address = START_ADDR_V + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    where denom1/denom2 are equal to 1 or 2 depending on the actual planar format, derived natively by hardware.

    buf_index is the index transmitted by the triggering module pointing to the first buffer of a frame.

Programming Restrictions

- For tiled address mode

  Image surface can only aligned to multiples of 256, thus the following restrictions.
  - START_ADDR, START_ADDR_U, START_ADDR_V need to be multiples of 256.
  - BUF_STRIDE, UV_BUF_STRIDE need to be multiples of 256
  - LINE_STRIDE, UV_LINE_STRIDE need to be multiples of 16
  - ADDR_H_OFFSET needs to be even in yuv planar format, or a multiple of bytes per pixel in other formats. If H_DIRECTION=DECREMENT, however, it should point to last valid byte, which is an odd offset..
  - ADDR_V_OFFSET needs to be multiple of 2 in YUV planar format, unless H_DIRECTION=DECREMENT, but with no restrictions on other color formats
- For linear address mode

  Image surface can be aligned 2, 4, or 8 bytes, depending on the color formats.

  As an additional restriction for display, START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16. When a surface is not aligned to 16 bytes, program START_ADDR with the memory surface address with its least 4 significant bits zeroed out and add these 4 address bits to the original H_OFFSET. (So the formulae in Starting Address Calculation above still hold)
  - For all formats:

    START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16.

    For 16-bpp formats,

    (START_ADDR+H_OFFSET) need to be a multiple of 2.

    For 32-bpp formats,

    (START_ADDR+H_OFFSET) needs to be a multiple of 4.
  - For yuv planar formats:

    BUF_STRIDE, UV_BUF_STRIDE:

    BUF_STRIDE[2:1]=UV_BUF_STRIDE[1:0]

    or as a stricter constraint: BUF_STRIDE be a multiple of 8, UV_BUF_STRIDE be a multiple of 4.

    LINE_STRIDE, UV_LINE_STRIDE:

    LINE_STRIDE and UV_LINE_STRIDE need to be at least 16.

    LINE_STRIDE needs to be a multiple of 8, UV_LINE_STRIDE needs to be a multiple of 4.

ADDR_H_OFFSET: Needs to be even unless H_DIRECTION=DECREMENT, in which case should point to the last byte pixel, which is at an odd offset.

ADDR_V_OFFSET: Needs to be even unless V_DIRECTION=DECREMENT, in which case should point to the last valid line, which is at an odd offset

- Memory allocation for non-host triggered case:

  When a window buffer is not controlled by host (software) then a frame may be stored in multiple buffers. In this case, the buffers must be contiguous in the memory because display will use the same luma or chroma line strides for all lines in the frame.

  Also buffer wraparound must not occur in the middle of the displayed part of the frame. The controlling module will send frame start and frame end indicators (flags) to display module to indicate the beginning and end of frame. Buffer start address is latched when frame start flag is active but the actual buffer start address is not switched until frame end flag is active. In the case where one buffer corresponds to one frame then frame start and frame end flag are active every time a buffer index is sent.

Offset: 0x800 | Byte Offset: 0x2000 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_START_ADDR: Window C Start Address. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

21.14.2 DC_WINBUF_C_START_ADDR_NS_0

Window C Shadowed Start Address

Offset: 0x801 | Byte Offset: 0x2004 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_START_ADDR_NS: Window C Shadowed Start Address. This is ARM set shadow of Start Address.

21.14.3 DC_WINBUF_C_START_ADDR_U_0

Window C Start Address for U plane

Offset: 0x802 | Byte Offset: 0x2008 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_START_ADDR_U: Window C Start Address for U plane. This is a byte address.

21.14.4 DC_WINBUF_C_START_ADDR_U_NS_0

Window C Shadowed Start Address for U plane

Offset: 0x803 | Byte Offset: 0x200c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_START_ADDR_U_NS: Window C Shadowed Start Address for U plane. This is ARM set shadow register of U start address

21.14.5 DC_WINBUF_C_START_ADDR_V_0

Window C Start Address for V plane

Offset: 0x804 | Byte Offset: 0x2010 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_START_ADDR_V: Window C Start Address for V plane. This is a byte address.

21.14.6 DC_WINBUF_C_START_ADDR_V_NS_0

Window C Shadowed Start Address for V plane

Offset: 0x805 | Byte Offset: 0x2014 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_START_ADDR_V_NS: Window C Shadowed Start Address for V plane. This is ARM set shadow register of U start address

21.14.7 DC_WINBUF_C_ADDR_H_OFFSET_0

Window C Horizontal address offset

Offset: 0x806 | Byte Offset: 0x2018 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_ADDR_H_OFFSET: Window C Horizontal address offset. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies horizontal offset of Y plane. The horizontal offsets of U/V plane is derived by hardware.

21.14.8 DC_WINBUF_C_ADDR_H_OFFSET_NS_0

Window C Shadowed Horizontal address offset

Offset: 0x807 | Byte Offset: 0x201c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_ADDR_H_OFFSET_NS: Window C Shadowed Horizontal address offset. This is ARM set shadow of ADDR_H_OFFSET

21.14.9 DC_WINBUF_C_ADDR_V_OFFSET_0

Window C Vertical address offset

Offset: 0x808 | Byte Offset: 0x2020 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_ADDR_V_OFFSET: Window C Vertical address offset. This is the vertical coordinate. For YUV planar pixel format, this specifies the vertical coordinate of Y plane. The vertical coordinate of U/V plane is derived by hardware.

21.14.10 DC_WINBUF_C_ADDR_V_OFFSET_NS_0

Window C Shadowed Vertical address offset

Offset: 0x809 | Byte Offset: 0x2024 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_ADDR_V_OFFSET_NS: Window C Shadowed Vertical address offset. This is ARM set shadow of ADDR_V_OFFSET

21.14.11 DC_WINBUF_C_UFLOW_STATUS_0

Window C FIFO Underflow Status Register

Offset: 0x80a | Byte Offset: 0x2028 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	COUNT_OFLOW: Flag bit that indicates that the underflow event counter has overflowed. There were too many events. If COUNT_OFLOW is set, UFLOW_COUNT is meaningless. Cleared on write.
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.

21.14.12 DC_WINBUF_C_SURFACE_KIND_0

Window C Surface Kind

Offset: 0x80b | Byte Offset: 0x202c | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxxxxxxxxxxxxxxxxx001xx00)

Bit	Reset	Description
6:4	0x1	C_BLOCK_HEIGHT: Block Height: For block linear, height of block in GOBs 0 = HEIGHT_1 1 = HEIGHT_2 2 = HEIGHT_4 3 = HEIGHT_8 4 = HEIGHT_16 5 = HEIGHT_32
1:0	0x0	C_SURFACE_KIND: Surface Kind: Pitched, Tiled, or Block Linear 0 = PITCH 1 = TILED 2 = BL_16B2

21.14.13 DC_WINBUF_C_SURFACE_WEIGHT_0

Window C Surface Weights

Offset: 0x80c | Byte Offset: 0x2030 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
6:5	X	C_SURFACE_WEIGHT_V: Window C V Surface Weights. 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16

Bit	Reset	Description
		0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
4:3	X	C_SURFACE_WEIGHT_U: Window C U or UV Surface Weights. 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16 0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
2:1	X	C_SURFACE_WEIGHT_Y: Window C Y or packed Surface Weights. 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16 0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
0	0x0	C_SURFACE_WEIGHT_OVERRIDE: Weight override 0 = DISABLE 1 = ENABLE

21.14.14 DC_WINBUF_C_START_ADDR_HI_0

Window C Higher 32 bits of Start Address

Makes the start address 64 bits by adding the higher 32 bits below.

Offset: 0x80d │ Byte Offset: 0x2034 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	C_START_ADDR_HI: Window C Start Address. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

21.14.15 DC_WINBUF_C_START_ADDR_HI_NS_0

Window C Shadowed Higher 32 bits of Start Address

Offset: 0x80e │ Byte Offset: 0x2038 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	C_START_ADDR_HI_NS: Window C Shadowed Start Address. This is ARM set shadow of Start Address.

21.14.16 DC_WINBUF_C_START_ADDR_HI_U_0

Window C Higher 32 bits of Start Address for U plane

Offset: 0x80f | Byte Offset: 0x203c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	C_START_ADDR_HI_U: Window C Start Address for U plane. This is a byte address.

21.14.17 DC_WINBUF_C_START_ADDR_HI_U_NS_0

Window C Shadowed Higher 32 bits of Start Address for U plane

Offset: 0x810 | Byte Offset: 0x2040 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	C_START_ADDR_HI_U_NS: Window C Shadowed Higher 32 bits of Start Address for U plane. This is ARM set shadow register of U start address

21.14.18 DC_WINBUF_C_START_ADDR_HI_V_0

Window C Higher 32 bits of Start Address for V plane

Offset: 0x811 | Byte Offset: 0x2044 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	C_START_ADDR_HI_V: Window C Higher 32 bits of Start Address for V plane. This is a byte address.

21.14.19 DC_WINBUF_C_START_ADDR_HI_V_NS_0

Window C Shadowed Higher 32 bits of Start Address for V plane

Offset: 0x812 | Byte Offset: 0x2048 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	C_START_ADDR_HI_V_NS: Window C Shadowed Higher 32 bits of Start Address for V plane. This is ARM set shadow register of U start address

21.14.20 DC_WINBUF_C_START_ADDR_FIELD2_0

Window C Start Address

Interlace/Stereo support. 64-bit base address for the odd field/right frame.

Offset: 0x813 | Byte Offset: 0x204c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_START_ADDR_FIELD2: Window C Start Address. This is a byte address. The LSB is not used for the 16-bpp non-planar pixel format, and the last 2 LSBs are not used for the 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

21.14.21 DC_WINBUF_C_START_ADDR_FIELD2_NS_0

Window C Shadowed Start Address

Offset: 0x814 | Byte Offset: 0x2050 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_START_ADDR_FIELD2_NS: Window C Shadowed Start Address. This is the ARM set shadow of the start address.

21.14.22 DC_WINBUF_C_START_ADDR_FIELD2_U_0

Window C Start Address for U plane

Offset: 0x815 | Byte Offset: 0x2054 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_START_ADDR_FIELD2_U: Window C Start Address for U plane. This is a byte address.

21.14.23 DC_WINBUF_C_START_ADDR_FIELD2_U_NS_0

Window C Shadowed Start Address for U plane

Offset: 0x816 | Byte Offset: 0x2058 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_START_ADDR_FIELD2_U_NS: Window C Shadowed Start Address for U plane. This is the ARM set shadow register of the U start address.

21.14.24 DC_WINBUF_C_START_ADDR_FIELD2_V_0

Window C Start Address for V plane

Offset: 0x817 | Byte Offset: 0x205c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_START_ADDR_FIELD2_V: Window C Start Address for V plane. This is a byte address.

21.14.25 DC_WINBUF_C_START_ADDR_FIELD2_V_NS_0

Window C Shadowed Start Address for V plane

Offset: 0x818 | Byte Offset: 0x2060 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_START_ADDR_FIELD2_V_NS: Window C Shadowed Start Address for V plane. This is the ARM set shadow register of the U start address.

21.14.26 DC_WINBUF_C_START_ADDR_FIELD2_HI_0

Window C Higher 32 bits of Start Address

Offset: 0x819 | Byte Offset: 0x2064 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	C_START_ADDR_FIELD2_HI: Window C Start Address. This is a byte address. The LSB is not used for the 16-bpp non-planar pixel format, and the last 2 LSBs are not used for the 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies the start address for the Y plane.

21.14.27 DC_WINBUF_C_START_ADDR_FIELD2_HI_NS_0

Window C Shadowed Higher 32 bits of Start Address

Offset: 0x81a | Byte Offset: 0x2068 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	C_START_ADDR_FIELD2_HI_NS: Window C Shadowed Start Address. This is the ARM set shadow of the start address.

21.14.28 DC_WINBUF_C_START_ADDR_FIELD2_HI_U_0

Window C Higher 32 bits of Start Address for U plane

Offset: 0x81b | Byte Offset: 0x206c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	C_START_ADDR_FIELD2_HI_U: Window C Start Address for U plane. This is a byte address.

21.14.29 DC_WINBUF_C_START_ADDR_FIELD2_HI_U_NS_0

Window C Shadowed Higher 32 bits of Start Address for U plane

Offset: 0x81c | Byte Offset: 0x2070 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	C_START_ADDR_FIELD2_HI_U_NS: Window C Shadowed Higher 32 bits of Start Address for U plane. This is the ARM set shadow register of the U start address.

21.14.30 DC_WINBUF_C_START_ADDR_FIELD2_HI_V_0

Window C Higher 32 bits of Start Address for V plane

Offset: 0x81d | Byte Offset: 0x2074 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	C_START_ADDR_FIELD2_HI_V: Window C Higher 32 bits of Start Address for V plane. This is a byte address.

21.14.31 DC_WINBUF_C_START_ADDR_FIELD2_HI_V_NS_0

Window C Shadowed Higher 32 bits of Start Address for V plane

Offset: 0x81e | Byte Offset: 0x2078 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	C_START_ADDR_FIELD2_HI_V_NS: Window C Shadowed Higher 32 bits of Start Address for V plane. This is the ARM set shadow register of the U start address.

21.14.32 DC_WINBUF_C_ADDR_H_OFFSET_FIELD2_0

Window C Horizontal address offset

Offset: 0x81f | Byte Offset: 0x207c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_ADDR_H_OFFSET_FIELD2: Window C Horizontal address offset. This is a byte address. The LSB is not used for the 16-bpp non-planar pixel format, and the last 2 LSBs are not used for the 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies the horizontal offset of the Y plane. The horizontal offsets of U/V plane are derived by hardware.

21.14.33 DC_WINBUF_C_ADDR_H_OFFSET_FIELD2_NS_0

Window C Shadowed Horizontal address offset

Offset: 0x820 | Byte Offset: 0x2080 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_ADDR_H_OFFSET_FIELD2_NS: Window C Shadowed Horizontal address offset. This is the ARM set shadow of ADDR_H_OFFSET.

21.14.34 DC_WINBUF_C_ADDR_V_OFFSET_FIELD2_0

Window C Vertical address offset

Offset: 0x821 | Byte Offset: 0x2084 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_ADDR_V_OFFSET_FIELD2: Window C Vertical address offset. This is the vertical coordinate. For YUV planar pixel format, this specifies the vertical coordinate of the Y plane. The vertical coordinate of U/V plane is derived by hardware.

21.14.35 DC_WINBUF_C_ADDR_V_OFFSET_FIELD2_NS_0

Window C Shadowed Vertical address offset

Offset: 0x822 | Byte Offset: 0x2088 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_ADDR_V_OFFSET_FIELD2_NS: Window C Shadowed Vertical address offset. This is the ARM set shadow of ADDR_V_OFFSET

21.14.36 DC_WINBUF_C_UFLOW_CTRL_0

When the underflow debug mode is enabled. Instead of sending out the last valid pixel, DBG_PIXEL is sent out.

Offset: 0x824 │ Byte Offset: 0x2090 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	C_UFLOW_CTRL_DBG_MODE: 0 = DISABLE 1 = ENABLE

21.14.37 DC_WINBUF_C_UFLOW_DBG_PIXEL_0

Offset: 0x825 │ Byte Offset: 0x2094 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_UFLOW_DBG_PIXEL

21.14.38 DC_WINBUF_C_UFLOW_THRESHOLD_0

Offset: 0x826 │ Byte Offset: 0x2098 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxx0000000000000)

Bit	Reset	Description
12:0	0x0	C_UFLOW_THRESHOLD

21.14.39 DC_WINBUF_C_SPOOL_UP_0

Spool up time configuration for different windows related logic.

SPOOL_UP_CTRL = MAX - This is the current Tegra K1 behavior. This allows for memfetch to start fetching for a window (irrespective of its relative position on the screen), at the very beginning of the VBlank of the next frame. Since the spool up time is maximum, this should have the least probability of underflow of the display pipe. However, on the down side, since all three windows would start fetching at the same time, and since display is a high priority client for the MC, it has a high potential to starve out other clients.

SPOOL_UP_EDGE :- By default the memfetch start signal generated by the hvtgen logic is one line clock wide.

For SPOOL_UP_CTRL = MAX:

- If SPOOL_UP_EDGE is programmed to NEGEDGE, it ensures that if any act_req is sent during the time vcounter = 0 (at the beginning of the frame). The activate value is loaded into the active state and the memfetch would see the latest updated value.

  The downside of programming SPOOL_UP_EDGE to NEGDGE is 1 line clock worth of spool up is lost. This is required based upon Tegra compatibility.
- If SPOOL_UP_EDGE is programmed to POSEDGE, the memfetch starts fetching at the very beginning of the frame start pulse (1 line clock wide) and gives maximum spool up time. However, this may give test failure and potential Tegra compatibility issues as described above.

Offset: 0x827 │ Byte Offset: 0x209c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
28:16	X	C_SPOOL_UP_DURATION
1	0x0	C_SPOOL_UP_EDGE: 0 = NEGEDGE 1 = POSEDGE

Bit	Reset	Description
0	0x0	C_SPOOL_UP_CTRL. This bit should be set to MAX for Tegra K1 devices. 0 = MAX 1 = PROGRAMMABLE

21.14.40 DC_WINBUF_C_SCALEFACTOR_THRESHOLD_0

Registers for latency allowance scaling. Threshold is in terms of 64 byte atoms. Display memfetch compares its buffer occupancy with the threshold and generates indication to MC if its buffer occupancy is above HWM or below LWM.

Offset: 0x828 │ Byte Offset: 0x20a0 │ Read/Write: R/W │ Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:16	0xffff	C_SF_LWM_THRESHOLD
15:0	0xffff	C_SF_HWM_THRESHOLD

21.14.41 DC_WINBUF_C_LATENCY_THRESHOLD_0

The LATENCY_THRESHOLD register controls the behavior of the rdy4_latency_event from the display to the MC.

If B_RDY4LATENCY_THRESHOLD_ENABLE is set to DISABLE, the sideband rdy4_latency_event is always asserted, indicating that the display is always ready for a DVFS event.

The B_RDY4LATENCY_THRESHOLD field indicates the required occupancy of line buffers of windows below which the sideband rdy4_latency_event is deasserted. The occupancy figure is in units of 64B atoms for memfetch windows and in units of scan lines for regular windows.

The B_RDY4LATENCY_SPOOLUP_DURATION field indicates the number of scan lines during spool-up when the sideband will remain asserted (DVFS is allowed). This takes affect only when B_RDY4LATENCY_SPOOLUP_CTRL is set to ALLOW.

If the B_RDY4LATENCY_SPOOLUP_CTRL field is set to DISALLOW, the above register field is not affected and DVFS is disallowed until the threshold is reached.

Offset: 0x829 │ Byte Offset: 0x20a4 │ Read/Write: R/W │ Reset: 0x0000ffff (0b00x00000000000001111111111111111)

Bit	Reset	Description
31	DISABLE	C_RDY4LATENCY_THRESHOLD_ENABLE: 0 = DISABLE 1 = ENABLE
30	DISALLOW	C_RDY4LATENCY_SPOOLUP_CTRL: 0 = DISALLOW 1 = ALLOW
28:16	0x0	C_RDY4LATENCY_SPOOLUP_DURATION
15:0	0xffff	C_RDY4LATENCY_THRESHOLD

21.14.42 DC_WINBUF_C_MEMFETCH_DEBUG_STATUS_0

Memfetch Debug Status

Offset: 0x82a │ Byte Offset: 0x20a8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
16	0x0	C_DEBUG_ENABLE: enable FGCG for flops added to debug signals in memfetch.
15	0x0	C_ALIGN_FIFO_NON_IDLE: align FIFO idle at end of frame. 0= align FIFO is idle at

Bit	Reset	Description
		end of frame. 1= align FIFO is not idle at end of frame
14	0x0	C_PIPE1_FIFO_NON_IDLE: pipe1 FIFO idle at end of frame. 0= pipe1 FIFO is idle at end of frame. 1= pipe1 FIFO is not idle at end of frame
13	0x0	C_PIPE0_FIFO_NON_IDLE: pipe0 FIFO idle at end of frame. 0= pipe0 FIFO is idle at end of frame. 1= pipe0 FIFO is not idle at end of frame
12	0x0	C_FRAME_FIFO_NON_IDLE: frame FIFO idle at end of frame. 0= frame FIFO is idle at end of frame. 1= frame FIFO is not idle at end of frame
11	0x0	C_SURFACE_FIFO_NON_IDLE: surface FIFO idle at end of frame. 0= surface FIFO is idle at end of frame. 1= surface FIFO is not idle at end of frame
10	0x0	C_TAG_FIFO_NON_IDLE: tag FIFO idle at end of frame. 0= tag FIFO is idle at end of frame. 1= tag FIFO is not idle at end of frame
9	0x0	C_RSP_FIFO_NON_IDLE: Response FIFO idle at end of frame. 0= response FIFO is idle at end of frame. 1= response FIFO is not idle at end of frame
8	0x0	C_REQ_FIFO_NON_IDLE: Request FIFO idle at end of frame. 0= request FIFO is idle at end of frame. 1= request FIFO is not idle at end of frame
7	0x0	C_DP_VALID_NON_IDLE: DP lines valid at end of frame. 0= DP lines are not valid at end of frame. 1= DP lines are valid at end of frame
6	0x0	C_V_SSM_NON_IDLE: V SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame
5	0x0	C_U_SSM_NON_IDLE: U SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame
4	0x0	C_Y_SSM_NON_IDLE: Y SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame
3	0x0	C_DP_POOL_AVAIL: DP thinks there is data in the pool at end of frame. 0= No pool data available at end of frame. 1= pool data available at end of frame
2	0x0	C_POOL_NOT_EMPTY: Status of pool at end of frame. 0= pool is empty at end of frame. 1= pool is not empty at end of frame
1	0x0	C_UNDERFLOW_LINE1: Underflow of line0. 0= No underflow. 1= line0 underflowed
0	0x0	C_UNDERFLOW_LINE0: Underflow of line0. 0= No underflow. 1= line0 underflowed

21.14.43 DC_WINBUF_C_MEMFETCH_CONTROL_0

Note: MEMFETCH_CLK_GATE_OVERRIDE should only be enabled AFTER the vertical scaler settings in the active copy - VDDA_INCREMENT is properly set up.

Memfetch Control Register

Offset: 0x82b │ Byte Offset: 0x20ac │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	SW Default	Description
1	0x0	NONE	C_MEMFETCH_CLK_GATE_OVERRIDE: Disable clock-gating of memfetch for a controller (all three windows) 0 = DISABLE 1 = ENABLE
0	0x0	ENABLE	C_MEMFETCH_RESET: Reset memfetch for a controller (all three windows) 0 = DISABLE 1 = ENABLE

21.14.44 DC_WINBUF_C_OCCUPANCY_THROTTLE_0

Offset: 0x82c | Byte Offset: 0x20b0 | Read/Write: R/W | Reset: 0xffff0000 (0b1111111111111111xxxxxxxxxxxxxxx0)

Bit	Reset	Description
31:16	0xffff	C_OCCUPANCY_MAX_THRESHOLD
0	0x0	C_OCCUPANCY_THROTTLE_MODE: 0 = DISABLE 1 = ENABLE

21.14.45 DC_WINBUF_C_SCRATCH_REGISTER_0_0

Offset: 0x82d | Byte Offset: 0x20b4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	C_SCRATCH_REGISTER_0: Scratch register 0

21.14.46 DC_WINBUF_C_SCRATCH_REGISTER_1_0

Offset: 0x82e | Byte Offset: 0x20b8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	C_SCRATCH_REGISTER_1: Scratch register 1

21.15 Window D (WIN_D) Registers

These registers control window D parameters.

The registers under DC_WIN are double buffered.

21.15.1 DC_WIN_D_WIN_OPTIONS_0

Window D Options

Class: Display Window Settings

Display Window D parameters

Offset: 0x700 | Byte Offset: 0x1c00 | Read/Write: R/W | Reset: 0x000000X0 (0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	0x0	D_WIN_ENABLE: Window D Window enable 0 = DISABLE 1 = ENABLE
6	X	D_COLOR_EXPAND: Window D 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

21.15.2 DC_WIN_D_COLOR_DEPTH_0

Window D Color Depth. For YCbCr data format, Cb and Cr are 8-bit unsigned values. For YUV data format, U and V are 8-bit signed values. YCbCr422R is similar to YCbCr422P but the Cb and Cr are shared vertically.

YUV422R is similar to YUV422P, but the U and V are shared vertically. YCbCr422RA is the same as YCbCr422R in memory and YUV422RA is the same as YUV422R in memory but while reading from memory, for YCbCr422RA and YUV422RA, chroma averaging is applied for each pixel pair so that they can be processed as YUV422 by the display pipeline.

For YCbCr422R and YUV422R, every other chroma pixels are not used (discarded) by the display pipeline. B6x2G6x2R6x2A8 is similar to B8G8R6A8 but with the 2 LSBs zeroed out. R6x2G6x2B6x2A8 is similar to R8G8B6A8 but with the 2 LSBs zeroed out.

Some formats have NVCF_ or T_ prefix aliases for NVidia surface format naming convention, wherein left-to-right component names correspond to MSB to LSB bits within a word. In old names without NVCF_ or T_, left-to-right component names generally map LSB to MSB within a word.

Offset: 0x703 │ Byte Offset: 0x1c0c │ Read/Write: R/W │ Reset: 0x0000000c (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0001100)

Bit	Reset	Description
6:0	B8G8R8A8	D_COLOR_DEPTH: Window D Color Depth. Supported color depths are: P8 = 8-bpp (palletized) B4G4R4A4 = 12-bpp B4G4R4 B5G5R5A = 15-bpp B5G5R5 AB5G5R5 = 15-bpp B5G5R5 B5G6R5 = 16-bpp B5G6R5 B8G8R8A8 = 32-bpp B8G8R8A8 R8G8B8A8 = 32-bpp R8G8B8A8 YCbCr422 = 16-bpp YCbCr422 packed YUV422 = 16-bpp YUV422 YCbCr420P = 16-bpp YCbCr420 planar YUV420P = 16-bpp YUV420 planar YCbCr422P = 16-bpp YCbCr422 planar YUV422P = 16-bpp YUV422 planar YCbCr422R = 16-bpp YCbCr422 rotated planar YUV422R = 16-bpp YUV422 rotated planar YCbCr422RA= 16-bpp YCbCr422 rotated planar with chroma averaging YUV422RA = 16-bpp YUV422 rotated planar with chroma averaging. Reserved values: 0,1,2,14,15,24,25. 3 = T_P8, P8 4 = T_A4R4G4B4, B4G4R4A4 5 = T_A1R5G5B5, B5G5R5A 6 = T_R5G6B5, B5G6R5 7 = T_R5G5B5A1, AB5G5R5 12 = T_A8R8G8B8, B8G8R8A8 13 = T_A8B8G8R8, R8G8B8A8 16 = T_U8_Y8__V8_Y8, CbYCrY422, YCbCr422 17 = T_U8_Y8__V8_Y8_TRUE, UYVY422, YUV422 18 = T_Y8___U8___V8_N420, YCbCr420P 19 = T_Y8___U8___V8_N420_TRUE, YUV420P 20 = T_Y8___U8___V8_N422, YCbCr422P 21 = T_Y8___U8___V8_N422_TRUE, YUV422P 22 = T_Y8___U8___V8_N422R, YCbCr422RP, YCbCr422R 23 = T_Y8___U8___V8_N422R_TRUE, YUV422RP, YUV422R 24 = T_V8_Y8__U8_Y8, CrYCbY422 25 = T_V8_Y8__U8_Y8_TRUE, VYUY422 26 = T_Y8, Y8 27 = T_A4B4G4R4, R4G4B4A4 28 = T_A1B5G5R5, R5G5B5A 29 = T_B5G5R5A1, AR5G5B5 30 = T_X1R5G5B5, B5G5R5X1 31 = T_R5G5B5X1, X1B5G5R5 32 = T_X1B5G5R5, R5G5B5X1 33 = T_B5G5R5X1, X1R5G5B5 34 = T_B5G6R5, R5G6B5 35 = T_B8G8R8A8 36 = T_R8G8B8A8 37 = T_X8R8G8B8, B8G8R8X8 38 = T_X8B8G8R8, R8G8B8X8 39 = T_A8Y8U8V8 40 = T_V8U8Y8A8 41 = T_Y8___U8___V8_N444, YCbCr444P 42 = T_Y8___U8V8_N420, YCrCb420SP 43 = T_Y8___V8U8_N420, YCbCr420SP 44 = T_Y8___U8V8_N422, YCrCb422SP 45 = T_Y8___V8U8_N422, YCbCr422SP 46 = T_Y8___U8V8_N422R, YCrCb422RSP 47 = T_Y8___V8U8_N422R, YCbCr422RSP 48 = T_Y8___U8V8_N444, YCrCb444SP

Bit	Reset	Description
		49 = T_Y8___V8U8_N444, YCbCr444SP 50 = T_A8Y8U8V8_TRUE 51 = T_V8U8Y8A8_TRUE 52 = T_Y8___U8___V8_N444_TRUE, YUV444P 53 = T_Y8___U8V8_N420_TRUE, YVU420SP 54 = T_Y8___V8U8_N420_TRUE, YUV420SP 55 = T_Y8___U8V8_N422_TRUE, YVU422SP 56 = T_Y8___V8U8_N422_TRUE, YUV422SP 57 = T_Y8___U8V8_N422R_TRUE, YVU422RSP 58 = T_Y8___V8U8_N422R_TRUE, YUV422RSP 59 = T_Y8___U8V8_N444_TRUE, YVU444SP 60 = T_Y8___V8U8_N444_TRUE, YUV444SP 61 = T_Y8_U8__Y8_V8, YCbYCr422 62 = T_Y8_U8__Y8_V8_TRUE, YUYV422 63 = T_Y8_V8__Y8_U8, YCrYCb422 64 = T_Y8_V8__Y8_U8_TRUE, YVYU422 65 = T_R8G8B8X8 66 = T_B8G8R8X8: new formats, reserved for future use

21.15.3 DC_WIN_D_POSITION_0

Window D Position

This register defines the H position and size of Window D after scaling (if there is any).

Offset: 0x704 │ Byte Offset: 0x1c10 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	D_V_POSITION: Window D V Position. This is specified with respect to the top edge of active display area.
12:0	X	D_H_POSITION: Window D H Position. This is specified with respect to the left edge of active display area.

21.15.4 DC_WIN_D_SIZE_0

Window D Size

This register defines the V position and size of Window D after scaling (if there is any).

Note: Programming on window size should guarantee the whole window is inside the active area, otherwise extra rows/columns will be fetched and discarded which affects performance.

Offset: 0x705 │ Byte Offset: 0x1c14 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	D_V_SIZE: Window D V Size (lines). This is the vertical size after scaling.
12:0	X	D_H_SIZE: Window D H Size (pixels). This is the horizontal size after scaling.

21.15.5 DC_WIN_D_LINE_STRIDE_0

Window D Line Stride

Offset: 0x70a | Byte Offset: 0x1c28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	D_LINE_STRIDE: Window D 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 D 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 in 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.

21.15.6 DC_WIN_D_BLEND_LAYER_CONTROL_0

Offset: 0x716 | Byte Offset: 0x1c58 | Read/Write: R/W | Reset: 0x01000000 (0bxxxxxxx10000000000000000000000)

Bit	Reset	Description
24	BLEND_ BYPASS	D_BLEND_BYPASS: 0 = BLEND_ENABLE 1 = BLEND_BYPASS
23:16	0x0	D_K2
15:8	0x0	D_K1
7:0	0x0	D_WINDOW_LAYER_DEPTH

21.15.7 DC_WIN_D_BLEND_MATCH_SELECT_0

Offset: 0x717 | Byte Offset: 0x1c5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xx00x000x000)

Bit	Reset	Description
13:12	0x0	D_BLEND_FACTOR_DST_ALPHA_MATCH_SELECT: 0 = ZERO 1 = ONE 2 = NEG_K1_TIMES_SRC 3 = K2
9:8	0x0	D_BLEND_FACTOR_SRC_ALPHA_MATCH_SELECT: 0 = ZERO 1 = K1 2 = K2
6:4	0x0	D_BLEND_FACTOR_DST_COLOR_MATCH_SELECT: 0 = ZERO 1 = ONE 2 = K1 3 = K2 4 = K1_TIMES_DST 5 = NEG_K1_TIMES_DST 6 = NEG_K1_TIMES_SRC 7 = NEG_K1
2:0	0x0	D_BLEND_FACTOR_SRC_COLOR_MATCH_SELECT:

Bit	Reset	Description
		0 = ZERO 1 = ONE 2 = K1 3 = K1_TIMES_DST 4 = NEG_K1_TIMES_DST 5 = K1_TIMES_SRC

21.15.8 DC_WIN_D_BLEND_ALPHA_1BIT_0

Offset: 0x719 | Byte Offset: 0x1c64 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	D_BLEND_WEIGHT1: This is used only for 1-bit alpha and alpha value of 1
7:0	0x0	D_BLEND_WEIGHT0: This is used only for 1-bit alpha and alpha value of 0.

21.16 WINBUF_D Registers

The registers under DC_WINBUF are triple-buffered.

21.16.1 DC_WINBUF_D_START_ADDR_0

Window D Start Address

Overview

START_ADDR, BUF_STRIDE, LINE_STRIDE, ADDR_H_OFFSET, ADDR_V_OFFSET combined, specify the starting address of a window buffer in the memory surface. Generally START_ADDR is programmed with the starting address of the memory surface. H/V_OFFSET specifies the address offsets of the pixel at the beginning of the window, with respect to the starting address of the memory surface. (There is an exception to that when memory surface is not well aligned, see “For linear address mode” under Programming Restrictions below.)

Note that "beginning of the window" here means the top-left corner of a window in normal mode, top-right corner in horizontal flipping, bottom-left corner in vertical flipping, and bottom-right in horizontal+vertical flipping.

Starting Address Calculation

These formulae are same for both tiled and linear modes. However, for linear mode, the addresses calculated are real physical addresses. For tiled mode, these addresses will be translated to the real physical addresses by the memory controller client before being used.

- When a window is host triggered, the starting address of a window is calculated as follows by hardware.
  - non-YUV-planar modes

    starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - YUV-planar modes

    Y-starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    U-starting-address = START_ADDR_U + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

V-starting-address = START_ADDR_V + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by hardware.

- When a window is non-host triggered, starting address of a window buffer is calculated as below.
  - non-YUV-planar mode

    starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - YUV-planar mode:

    Y-starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    U-starting-address = START_ADDR_U + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    V-starting-address = START_ADDR_V + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    where denom1/denom2 are equal to 1 or 2 depending on the actual planar format, derived natively by hardware.

    buf_index is the index transmitted by the triggering module pointing to the first buffer of a frame.

Programming Restrictions

- For tiled address mode

  Image surface can only aligned to multiples of 256, thus the following restrictions.
  - START_ADDR, START_ADDR_U, START_ADDR_V need to be multiples of 256.
  - BUF_STRIDE, UV_BUF_STRIDE need to be multiples of 256
  - LINE_STRIDE, UV_LINE_STRIDE need to be multiples of 16
  - ADDR_H_OFFSET needs to be even in yuv planar format, or a multiple of bytes per pixel in other formats. If H_DIRECTION=DECREMENT, however, it should point to last valid byte, which is an odd offset..
  - ADDR_V_OFFSET needs to be multiple of 2 in YUV planar format, unless H_DIRECTION=DECREMENT, but with no restrictions on other color formats
- For linear address mode

  Image surface can be aligned 2, 4, or 8 bytes, depending on the color formats.

  As an additional restriction for display, START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16. When a surface is not aligned to 16 bytes, program START_ADDR with the memory surface address with its least 4 significant bits zeroed out and add these 4 address bits to the original H_OFFSET. (So the formulae in Starting Address Calculation above still hold)
  - For all formats:

    START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16.

    For 16-bpp formats,

    (START_ADDR+H_OFFSET) need to be a multiple of 2.

    For 32-bpp formats,

    (START_ADDR+H_OFFSET) needs to be a multiple of 4.
  - For yuv planar formats:

    BUF_STRIDE, UV_BUF_STRIDE:

    BUF_STRIDE[2:1]=UV_BUF_STRIDE[1:0]

    or as a stricter constraint: BUF_STRIDE be a multiple of 8, UV_BUF_STRIDE be a multiple of 4.

LINE_STRIDE, UV_LINE_STRIDE:

LINE_STRIDE and UV_LINE_STRIDE need to be at least 16.

LINE_STRIDE needs to be a multiple of 8, UV_LINE_STRIDE needs to be a multiple of 4.

ADDR_H_OFFSET: Needs to be even unless H_DIRECTION=DECREMENT, in which case should point to the last byte pixel, which is at an odd offset.

ADDR_V_OFFSET: Needs to be even unless V_DIRECTION=DECREMENT, in which case should point to the last valid line, which is at an odd offset

- Memory allocation for non-host triggered case:

  When a window buffer is not controlled by host (software) then a frame may be stored in multiple buffers. In this case, the buffers must be contiguous in the memory because display will use the same luma or chroma line strides for all lines in the frame.

  Also buffer wraparound must not occur in the middle of the displayed part of the frame. The controlling module will send frame start and frame end indicators (flags) to display module to indicate the beginning and end of frame. Buffer start address is latched when frame start flag is active but the actual buffer start address is not switched until frame end flag is active. In the case where one buffer corresponds to one frame then frame start and frame end flag are active every time a buffer index is sent.

Offset: 0x800 │ Byte Offset: 0x2000 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	D_START_ADDR: Window D 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.

21.16.2 DC_WINBUF_D_START_ADDR_NS_0

Window D Shadowed Start Address

Offset: 0x801 │ Byte Offset: 0x2004 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	D_START_ADDR_NS: Window D Shadowed Start Address. This is ARM set shadow of Start Address.

21.16.3 DC_WINBUF_D_ADDR_H_OFFSET_0

Window D Horizontal address offset

Offset: 0x806 │ Byte Offset: 0x2018 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	D_ADDR_H_OFFSET: Window D Horizontal address offset. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies horizontal offset of Y plane. The horizontal offsets of U/V plane is derived by hardware.

21.16.4 DC_WINBUF_D_ADDR_H_OFFSET_NS_0

Window D Shadowed Horizontal address offset

Offset: 0x807 | Byte Offset: 0x201c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	D_ADDR_H_OFFSET_NS: Window D Shadowed Horizontal address offset. This is ARM set shadow of ADDR_H_OFFSET

21.16.5 DC_WINBUF_D_ADDR_V_OFFSET_0

Window D Vertical address offset

Offset: 0x808 | Byte Offset: 0x2020 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	D_ADDR_V_OFFSET: Window D Vertical address offset. This is the vertical coordinate. For YUV planar pixel format, this specifies the vertical coordinate of Y plane. The vertical coordinate of U/V plane is derived by hardware.

21.16.6 DC_WINBUF_D_ADDR_V_OFFSET_NS_0

Window D Shadowed Vertical address offset

Offset: 0x809 | Byte Offset: 0x2024 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	D_ADDR_V_OFFSET_NS: Window D Shadowed Vertical address offset. This is ARM set shadow of ADDR_V_OFFSET

21.16.7 DC_WINBUF_D_UFLOW_STATUS_0

Window D FIFO Underflow Status Register

Offset: 0x80a | Byte Offset: 0x2028 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	COUNT_OFLOW: Flag bit that indicates that the underflow event counter has overflowed. There were too many events. If COUNT_OFLOW is set, UFLOW_COUNT is meaningless. Cleared on write.
23:0	X	UFLOW_COUNT: Underflow count. This field indicates the number of contiguous groups of output pixels for which there was no data in the FIFO. For example, if the valid from the FIFO is low for 10 consecutive cycles and then goes high, the counter will increment by one. Reset to zero on write.

21.16.8 DC_WINBUF_D_START_ADDR_HI_0

Window D Higher 32 bits of Start Address

Makes the start address 64 bits by adding the higher 32 bits below.

Offset: 0x80d | Byte Offset: 0x2034 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	D_START_ADDR_HI: Window D 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.

21.16.9 DC_WINBUF_D_START_ADDR_HI_NS_0

Window D Shadowed Higher 32 bits of Start Address

Offset: 0x80e │ Byte Offset: 0x2038 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	D_START_ADDR_HI_NS: Window D Shadowed Start Address. This is ARM set shadow of Start Address.

21.16.10 DC_WINBUF_D_UFLOW_CTRL_0

When the underflow debug mode is enabled. Instead of sending out the last valid pixel, DBG_PIXEL is sent out.

If D_UFLOW_CYA is set, the underflow recovery logic for simple windows is bypassed and replaced with an oflow-pop logic present in Tegra 3 styled windows.

Offset: 0x824 │ Byte Offset: 0x2090 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	D_UFLOW_CYA: 0 = DISABLE 1 = ENABLE
0	0x0	D_UFLOW_CTRL_DBG_MODE: 0 = DISABLE 1 = ENABLE

21.16.11 DC_WINBUF_D_UFLOW_DBG_PIXEL_0

Offset: 0x825 │ Byte Offset: 0x2094 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	D_UFLOW_DBG_PIXEL

21.16.12 DC_WINBUF_D_SPOOL_UP_0

Spool up time configuration for different windows related logic.

SPOOL_UP_CTRL = MAX - This is the current Tegra K1 behavior. This allows for memfetch to start fetching for a window (irrespective of its relative position on the screen), at the very beginning of the VBlank of the next frame. Since the spool up time is maximum, this should have the least probability of underflow of the display pipe. However, on the down side, since all three windows would start fetching at the same time, and since display is a high priority client for the MC, it has a high potential to starve out other clients.

SPOOL_UP_EDGE :- By default the memfetch start signal generated by the hvtgen logic is one line clock wide.

For SPOOL_UP_CTRL = MAX:

- If SPOOL_UP_EDGE is programmed to NEGEDGE, it ensures that if any act_req is sent during the time vcounter = 0 (at the beginning of the frame). The activate value is loaded into the active state and the memfetch would see the latest updated value.

The downside of programming SPOOL_UP_EDGE to NEGDGE is 1 line clock worth of spool up is lost. This is required based upon Tegra compatibility.

- If SPOOL_UP_EDGE is programmed to POSEDGE, the memfetch starts fetching at the very beginning of the frame start pulse (1 line clock wide) and gives maximum spool up time. However, this may give test failure and potential Tegra compatibility issues as described above.

Offset: 0x827 | Byte Offset: 0x209c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
28:16	X	D_SPOOL_UP_DURATION
1	0x0	D_SPOOL_UP_EDGE: 0 = NEGEDGE 1 = POSEDGE
0	0x0	D_SPOOL_UP_CTRL. 0 = MAX 1 = PROGRAMMABLE

21.16.13 DC_WINBUF_D_LATENCY_THRESHOLD_0

Offset: 0x829 | Byte Offset: 0x20a4 | Read/Write: R/W | Reset: 0x0000ffff (0b00x00000000000001111111111111111)

Bit	Reset	Description
31	DISABLE	D_RDY4LATENCY_THRESHOLD_ENABLE: 0 = DISABLE 1 = ENABLE
30	DISALLOW	D_RDY4LATENCY_SPOOLUP_CTRL: 0 = DISALLOW 1 = ALLOW
28:16	0x0	D_RDY4LATENCY_SPOOLUP_DURATION
15:0	0xffff	D_RDY4LATENCY_THRESHOLD

21.16.14 DC_WINBUF_D_MEMFETCH_DEBUG_STATUS_0

Memfetch Debug Status

Offset: 0x82a | Byte Offset: 0x20a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx00xxxxxxx0)

Bit	Reset	Description
9	0x0	D_RSP_FIFO_NON_IDLE: Response FIFO idle at end of frame. 0= response FIFO is idle at end of frame. 1= response FIFO is not idle at end of frame
8	0x0	D_REQ_FIFO_NON_IDLE: Request FIFO idle at end of frame. 0= request FIFO is idle at end of frame. 1= request FIFO is not idle at end of frame
0	0x0	D_UNDERFLOW_LINE0: Underflow of line0. 0= No underflow. 1= line0 underflowed

21.16.15 DC_WINBUF_D_MEMFETCH_CONTROL_0

Memfetch Control Register

Offset: 0x82b | Byte Offset: 0x20ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	D_MEMFETCH_CLK_GATE_OVERRIDE: Disable clock-gating of memfetch for a controller (all three windows)

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
0	0x0	D_MEMFETCH_RESET: Reset memfetch for a controller (all three windows) 0 = DISABLE 1 = ENABLE

21.16.16 DC_WINBUF_D_SCRATCH_REGISTER_0_0

Offset: 0x82d │ Byte Offset: 0x20b4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	D_SCRATCH_REGISTER_0: Scratch register 0

21.16.17 DC_WINBUF_D_SCRATCH_REGISTER_1_0

Offset: 0x82e │ Byte Offset: 0x20b8 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	D_SCRATCH_REGISTER_1: Scratch register 1

21.17 Window T (WIN_T) Registers

These registers control window T parameters.

The registers under DC_WIN are double buffered.

21.17.1 DC_WIN_T_WIN_OPTIONS_0

Window T Options

Class: Display Window Settings

Display Window T parameters

Offset: 0x700 │ Byte Offset: 0x1c00 │ Read/Write: R/W │ Reset: 0x000000X0 (0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	0x0	T_WIN_ENABLE: Window T Window enable 0 = DISABLE 1 = ENABLE
6	X	T_COLOR_EXPAND: Window T 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

21.17.2 DC_WIN_T_COLOR_DEPTH_0

Window T Color Depth. For YCbCr data format, Cb and Cr are 8-bit unsigned values. For YUV data format, U and V are 8-bit signed values. YCbCr422R is similar to YCbCr422P but the Cb and Cr are shared vertically.

YUV422R is similar to YUV422P, but the U and V are shared vertically. YCbCr422RA is the same as YCbCr422R in memory and YUV422RA is the same as YUV422R in memory but while reading from memory, for YCbCr422RA and YUV422RA, chroma averaging is applied for each pixel pair so that they can be processed as YUV422 by the display pipeline.

For YCbCr422R and YUV422R, every other chroma pixels are not used (discarded) by the display pipeline. B6x2G6x2R6x2A8 is similar to B8G8R6A8 but with the 2 LSBs zeroed out. R6x2G6x2B6x2A8 is similar to R8G8B6A8 but with the 2 LSBs zeroed out.

Some formats have NVCF_ or T_ prefix aliases for NVidia surface format naming convention, wherein left-to-right component names correspond to MSB to LSB bits within a word. In old names without NVCF_ or T_, left-to-right component names generally map LSB to MSB within a word.

Offset: 0x703 │ Byte Offset: 0x1c0c │ Read/Write: R/W │ Reset: 0x0000000c (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0001100)

Bit	Reset	Description
6:0	B8G8R8A8	T_COLOR_DEPTH: Window T Color Depth. Supported color depths are: P8 = 8-bpp (palletized) B4G4R4A4 = 12-bpp B4G4R4 B5G5R5A = 15-bpp B5G5R5 AB5G5R5 = 15-bpp B5G5R5 B5G6R5 = 16-bpp B5G6R5 B8G8R8A8 = 32-bpp B8G8R8A8 R8G8B8A8 = 32-bpp R8G8B8A8 YCbCr422 = 16-bpp YCbCr422 packed YUV422 = 16-bpp YUV422 YCbCr420P = 16-bpp YCbCr420 planar YUV420P = 16-bpp YUV420 planar YCbCr422P = 16-bpp YCbCr422 planar YUV422P = 16-bpp YUV422 planar YCbCr422R = 16-bpp YCbCr422 rotated planar YUV422R = 16-bpp YUV422 rotated planar YCbCr422RA= 16-bpp YCbCr422 rotated planar with chroma averaging YUV422RA = 16-bpp YUV422 rotated planar with chroma averaging. Reserved values: 0,1,2,14,15,24,25. 3 = T_P8, P8 4 = T_A4R4G4B4, B4G4R4A4 5 = T_A1R5G5B5, B5G5R5A 6 = T_R5G6B5, B5G6R5 7 = T_R5G5B5A1, AB5G5R5 12 = T_A8R8G8B8, B8G8R8A8 13 = T_A8B8G8R8, R8G8B8A8 16 = T_U8_Y8__V8_Y8, CbYCrY422, YCbCr422 17 = T_U8_Y8__V8_Y8_TRUE, UYVY422, YUV422 18 = T_Y8___U8___V8_N420, YCbCr420P 19 = T_Y8___U8___V8_N420_TRUE, YUV420P 20 = T_Y8___U8___V8_N422, YCbCr422P 21 = T_Y8___U8___V8_N422_TRUE, YUV422P 22 = T_Y8___U8___V8_N422R, YCbCr422RP, YCbCr422R 23 = T_Y8___U8___V8_N422R_TRUE, YUV422RP, YUV422R 24 = T_V8_Y8__U8_Y8, CrYCbY422 25 = T_V8_Y8__U8_Y8_TRUE, VYUY422 26 = T_Y8, Y8 27 = T_A4B4G4R4, R4G4B4A4 28 = T_A1B5G5R5, R5G5B5A 29 = T_B5G5R5A1, AR5G5B5 30 = T_X1R5G5B5, B5G5R5X1 31 = T_R5G5B5X1, X1B5G5R5 32 = T_X1B5G5R5, R5G5B5X1 33 = T_B5G5R5X1, X1R5G5B5 34 = T_B5G6R5, R5G6B5 35 = T_B8G8R8A8 36 = T_R8G8B8A8 37 = T_X8R8G8B8, B8G8R8X8 38 = T_X8B8G8R8, R8G8B8X8 39 = T_A8Y8U8V8 40 = T_V8U8Y8A8 41 = T_Y8___U8___V8_N444, YCbCr444P

Bit	Reset	Description
		42 = T_Y8___U8V8_N420, YCrCb420SP 43 = T_Y8___V8U8_N420, YCbCr420SP 44 = T_Y8___U8V8_N422, YCrCb422SP 45 = T_Y8___V8U8_N422, YCbCr422SP 46 = T_Y8___U8V8_N422R, YCrCb422RSP 47 = T_Y8___V8U8_N422R, YCbCr422RSP 48 = T_Y8___U8V8_N444, YCrCb444SP 49 = T_Y8___V8U8_N444, YCbCr444SP 50 = T_A8Y8U8V8_TRUE 51 = T_V8U8Y8A8_TRUE 52 = T_Y8___U8___V8_N444_TRUE, YUV444P 53 = T_Y8___U8V8_N420_TRUE, YVU420SP 54 = T_Y8___V8U8_N420_TRUE, YUV420SP 55 = T_Y8___U8V8_N422_TRUE, YVU422SP 56 = T_Y8___V8U8_N422_TRUE, YUV422SP 57 = T_Y8___U8V8_N422R_TRUE, YVU422RSP 58 = T_Y8___V8U8_N422R_TRUE, YUV422RSP 59 = T_Y8___U8V8_N444_TRUE, YVU444SP 60 = T_Y8___V8U8_N444_TRUE, YUV444SP 61 = T_Y8_U8__Y8_V8, YCbYCr422 62 = T_Y8_U8__Y8_V8_TRUE, YUYV422 63 = T_Y8_V8__Y8_U8, YCrYCb422 64 = T_Y8_V8__Y8_U8_TRUE, YVYU422 65 = T_R8G8B8X8 66 = T_B8G8R8X8: new formats, reserved for future use

21.17.3 DC_WIN_T_POSITION_0

Window T Position

This register defines the H position and size of Window T after scaling (if there is any).

Offset: 0x704 │ Byte Offset: 0x1c10 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	T_V_POSITION: Window T V Position. This is specified with respect to the top edge of active display area.
12:0	X	T_H_POSITION: Window T H Position. This is specified with respect to the left edge of active display area.

21.17.4 DC_WIN_T_SIZE_0

Window T Size

This register defines the V position and size of Window T after scaling (if there is any).

> **Note:** Programming on window size should guarantee the whole window is inside the active area, otherwise extra rows/columns will be fetched and discarded which affects performance.

Offset: 0x705 | Byte Offset: 0x1c14 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	T_V_SIZE: Window T V Size (lines). This is the vertical size after scaling.
12:0	X	T_H_SIZE: Window T H Size (pixels). This is the horizontal size after scaling.

21.17.5 DC_WIN_T_LINE_STRIDE_0

Window T Line Stride

Offset: 0x70a | Byte Offset: 0x1c28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	T_LINE_STRIDE: Window T 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 T 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 in 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.

21.17.6 DC_WIN_T_BLEND_LAYER_CONTROL_0

Offset: 0x716 | Byte Offset: 0x1c58 | Read/Write: R/W | Reset: 0x01000000 (0bxxxxxxx10000000000000000000000)

Bit	Reset	Description
24	BLEND_ BYPASS	T_BLEND_BYPASS: 0 = BLEND_ENABLE 1 = BLEND_BYPASS
23:16	0x0	T_K2
15:8	0x0	T_K1
7:0	0x0	T_WINDOW_LAYER_DEPTH

21.17.7 DC_WIN_T_BLEND_MATCH_SELECT_0

Offset: 0x717 | Byte Offset: 0x1c5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xx00x000x000)

Bit	Reset	Description
13:12	0x0	T_BLEND_FACTOR_DST_ALPHA_MATCH_SELECT: 0 = ZERO 1 = ONE 2 = NEG_K1_TIMES_SRC 3 = K2
9:8	0x0	T_BLEND_FACTOR_SRC_ALPHA_MATCH_SELECT: 0 = ZERO 1 = K1 2 = K2
6:4	0x0	T_BLEND_FACTOR_DST_COLOR_MATCH_SELECT: 0 = ZERO 1 = ONE 2 = K1 3 = K2 4 = K1_TIMES_DST 5 = NEG_K1_TIMES_DST 6 = NEG_K1_TIMES_SRC 7 = NEG_K1
2:0	0x0	T_BLEND_FACTOR_SRC_COLOR_MATCH_SELECT: 0 = ZERO 1 = ONE 2 = K1 3 = K1_TIMES_DST 4 = NEG_K1_TIMES_DST 5 = K1_TIMES_SRC

21.17.8 DC_WIN_T_BLEND_ALPHA_1BIT_0

Offset: 0x719 | Byte Offset: 0x1c64 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	T_BLEND_WEIGHT1: This is used only for 1-bit alpha and alpha value of 1
7:0	0x0	T_BLEND_WEIGHT0: This is used only for 1-bit alpha and alpha value of 0.

21.18 WINBUF_T Registers

The registers under DC_WINBUF are triple-buffered.

21.18.1 DC_WINBUF_T_START_ADDR_0

Window T Start Address

Overview

START_ADDR, BUF_STRIDE, LINE_STRIDE, ADDR_H_OFFSET, ADDR_V_OFFSET combined, specify the starting address of a window buffer in the memory surface. Generally, START_ADDR is programmed with the starting address of the memory surface. H/V_OFFSET specifies the address offsets of the pixel at the beginning of the window, with respect to the starting address of the memory surface. (There is an exception to that when memory surface is not well aligned, see “For linear address mode” under Programming Restrictions below.)

Note that "beginning of the window" here means the top-left corner of a window in normal mode, top-right corner in horizontal flipping, bottom-left corner in vertical flipping, and bottom-right in horizontal+vertical flipping.

Starting Address Calculation

These formulae are same for both tiled and linear modes. However, for linear mode, the addresses calculated are real physical addresses. For tiled mode, these addresses will be translated to the real physical addresses by the memory controller client before being used.

- When a window is host triggered, the starting address of a window is calculated as follows by hardware.
  - non-YUV-planar modes

    starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - YUV-planar modes

    Y-starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    U-starting-address = START_ADDR_U + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    V-starting-address = START_ADDR_V + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by hardware.
- When a window is non-host triggered, starting address of a window buffer is calculated as below.
  - non-YUV-planar mode

    starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - YUV-planar mode:

    Y-starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    U-starting-address = START_ADDR_U + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    V-starting-address = START_ADDR_V + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    where denom1/denom2 are equal to 1 or 2 depending on the actual planar format, derived natively by hardware.

    buf_index is the index transmitted by the triggering module pointing to the first buffer of a frame.

Programming Restrictions

- For tiled address mode

  Image surface can only aligned to multiples of 256, thus the following restrictions.

  - START_ADDR, START_ADDR_U, START_ADDR_V need to be multiples of 256.
  - BUF_STRIDE, UV_BUF_STRIDE need to be multiples of 256
  - LINE_STRIDE, UV_LINE_STRIDE need to be multiples of 16
  - ADDR_H_OFFSET needs to be even in yuv planar format, or a multiple of bytes per pixel in other formats. If H_DIRECTION=DECREMENT, however, it should point to last valid byte, which is an odd offset..
  - ADDR_V_OFFSET needs to be multiple of 2 in YUV planar format, unless H_DIRECTION=DECREMENT, but with no restrictions on other color formats

- For linear address mode

  Image surface can be aligned 2, 4, or 8 bytes, depending on the color formats.

  As an additional restriction for display, START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16. When a surface is not aligned to 16 bytes, program START_ADDR with the memory surface address with its least 4 significant bits zeroed out and add these 4 address bits to the original H_OFFSET. (So the formulae in Starting Address Calculation above still hold)

  - For all formats:

    START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16.

    For 16-bpp formats,

    (START_ADDR+H_OFFSET) need to be a multiple of 2.

    For 32-bpp formats,

    (START_ADDR+H_OFFSET) needs to be a multiple of 4.

  - For YUV planar formats:

    BUF_STRIDE, UV_BUF_STRIDE:

    BUF_STRIDE[2:1]=UV_BUF_STRIDE[1:0]

    or as a stricter constraint: BUF_STRIDE be a multiple of 8, UV_BUF_STRIDE be a multiple of 4.

    LINE_STRIDE, UV_LINE_STRIDE:

    LINE_STRIDE and UV_LINE_STRIDE need to be at least 16.

    LINE_STRIDE needs to be a multiple of 8, UV_LINE_STRIDE needs to be a multiple of 4.

    ADDR_H_OFFSET: Needs to be even unless H_DIRECTION=DECREMENT, in which case should point to the last byte pixel, which is at an odd offset.

    ADDR_V_OFFSET: Needs to be even unless V_DIRECTION=DECREMENT, in which case should point to the last valid line, which is at an odd offset

- Memory allocation for non-host triggered case:

  When a window buffer is not controlled by host (software) then a frame may be stored in multiple buffers. In this case, the buffers must be contiguous in the memory because display will use the same luma or chroma line strides for all lines in the frame.

  Also buffer wraparound must not occur in the middle of the displayed part of the frame. The controlling module will send frame start and frame end indicators (flags) to display module to indicate the beginning and end of frame. Buffer start address is latched when frame start flag is active but the actual buffer start address is not switched until frame end flag is active. In the case where one buffer corresponds to one frame then frame start and frame end flag are active every time a buffer index is sent.

Offset: 0x800 | Byte Offset: 0x2000 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	T_START_ADDR: Window T 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.

21.18.2 DC_WINBUF_T_START_ADDR_NS_0

Window T Shadowed Start Address

Offset: 0x801 | Byte Offset: 0x2004 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	T_START_ADDR_NS: Window H Shadowed Start Address. This is ARM set shadow of Start Address.

21.18.3 DC_WINBUF_T_ADDR_H_OFFSET_0

Window T Horizontal address offset

Offset: 0x806 | Byte Offset: 0x2018 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	T_ADDR_H_OFFSET: Window T Horizontal address offset. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies horizontal offset of Y plane. The horizontal offsets of U/V plane is derived by hardware.

21.18.4 DC_WINBUF_T_ADDR_H_OFFSET_NS_0

Window T Shadowed Horizontal address offset

Offset: 0x807 | Byte Offset: 0x201c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	T_ADDR_H_OFFSET_NS: Window T Shadowed Horizontal address offset. This is ARM set shadow of ADDR_H_OFFSET

21.18.5 DC_WINBUF_T_ADDR_V_OFFSET_0

Window T Vertical address offset

Offset: 0x808 | Byte Offset: 0x2020 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	T_ADDR_V_OFFSET: Window T Vertical address offset. This is the vertical coordinate. For YUV planar pixel format, this specifies the vertical coordinate of Y plane. The vertical coordinate of U/V plane is derived by hardware.

21.18.6 DC_WINBUF_T_ADDR_V_OFFSET_NS_0

Window T Shadowed Vertical address offset

Offset: 0x809 | Byte Offset: 0x2024 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	T_ADDR_V_OFFSET_NS: Window T Shadowed Vertical address offset. This is ARM set shadow of ADDR_V_OFFSET

21.18.7 DC_WINBUF_T_UFLOW_STATUS_0

Window T FIFO Underflow Status Register

Offset: 0x80a | Byte Offset: 0x2028 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	COUNT_OFLOW: Flag bit that indicates that the underflow event counter has overflowed. There were too many events. If COUNT_OFLOW is set, UFLOW_COUNT is meaningless. Cleared on write.
23:0	X	UFLOW_COUNT: Underflow count. This field indicates the number of contiguous groups of output pixels for which there was no data in the FIFO. For example, if the valid from the FIFO is low for 10 consecutive cycles and then goes high, the counter will increment by one. Reset to zero on write.

21.18.8 DC_WINBUF_T_START_ADDR_HI_0

Window T Higher 32 bits of Start Address

Makes the start address 64 bits by adding the higher 32 bits below.

Offset: 0x80d | Byte Offset: 0x2034 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	T_START_ADDR_HI: Window T 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.

21.18.9 DC_WINBUF_T_START_ADDR_HI_NS_0

Window T Shadowed Higher 32 bits of Start Address

Offset: 0x80e | Byte Offset: 0x2038 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	T_START_ADDR_HI_NS: Window T Shadowed Start Address. This is ARM set shadow of Start Address.

21.18.10 DC_WINBUF_T_UFLOW_CTRL_0

When the underflow debug mode is enabled. Instead of sending out the last valid pixel, DBG_PIXEL is sent out.

If T_UFLOW_CYA is set, the underflow recovery logic for simple windows is bypassed and replaced with an oflow-pop logic present in Tegra 3 styled windows.

Offset: 0x824 │ Byte Offset: 0x2090 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	T_UFLOW_CYA: 0 = DISABLE 1 = ENABLE
0	0x0	T_UFLOW_CTRL_DBG_MODE: 0 = DISABLE 1 = ENABLE

21.18.11 DC_WINBUF_T_UFLOW_DBG_PIXEL_0

Offset: 0x825 │ Byte Offset: 0x2094 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	T_UFLOW_DBG_PIXEL

21.18.12 DC_WINBUF_T_SPOOL_UP_0

Spool up time configuration for different windows related logic.

SPOOL_UP_CTRL = MAX - This is the current Tegra K1 behavior. This allows for memfetch to start fetching for a window (irrespective of its relative position on the screen), at the very beginning of the VBlank of the next frame. Since the spool up time is maximum, this should have the least probability of underflow of the display pipe. However, on the down side, since all three windows would start fetching at the same time, and since display is a high priority client for the MC, it has a high potential to starve out other clients.

SPOOL_UP_EDGE :- By default the memfetch start signal generated by the hvtgen logic is one line clock wide.

For SPOOL_UP_CTRL = MAX:

- If SPOOL_UP_EDGE is programmed to NEGEDGE, it ensures that if any act_req is sent during the time vcounter = 0 (at the beginning of the frame). The activate value is loaded into the active state and the memfetch would see the latest updated value.

  The downside of programming SPOOL_UP_EDGE to NEGDGE is 1 line clock worth of spool up is lost. This is required based upon Tegra compatibility.
- If SPOOL_UP_EDGE is programmed to POSEDGE, the memfetch starts fetching at the very beginning of the frame start pulse (1 line clock wide) and gives maximum spool up time. However, this may give test failure and potential Tegra compatibility issues as described above.

Offset: 0x827 │ Byte Offset: 0x209c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
28:16	X	T_SPOOL_UP_DURATION
1	0x0	T_SPOOL_UP_EDGE: 0 = NEGEDGE 1 = POSEDGE

Bit	Reset	Description
0	0x0	T_SPOOL_UP_CTRL. 0 = MAX 1 = PROGRAMMABLE

21.18.13 DC_WINBUF_T_LATENCY_THRESHOLD_0

Offset: 0x829 | Byte Offset: 0x20a4 | Read/Write: R/W | Reset: 0x0000ffff (0b00x00000000000001111111111111111)

Bit	Reset	Description
31	DISABLE	T_RDY4LATENCY_THRESHOLD_ENABLE: 0 = DISABLE 1 = ENABLE
30	DISALLOW	T_RDY4LATENCY_SPOOLUP_CTRL: 0 = DISALLOW 1 = ALLOW
28:16	0x0	T_RDY4LATENCY_SPOOLUP_DURATION
15:0	0xffff	T_RDY4LATENCY_THRESHOLD

21.18.14 DC_WINBUF_T_MEMFETCH_DEBUG_STATUS_0

Memfetch Debug Status

Offset: 0x82a | Byte Offset: 0x20a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx00xxxxxxx0)

Bit	Reset	Description
9	0x0	T_RSP_FIFO_NON_IDLE: Response FIFO idle at end of frame. 0= response FIFO is idle at end of frame. 1= response FIFO is not idle at end of frame
8	0x0	T_REQ_FIFO_NON_IDLE: Request FIFO idle at end of frame. 0= request FIFO is idle at end of frame. 1= request FIFO is not idle at end of frame
0	0x0	T_UNDERFLOW_LINE0: Underflow of line0. 0= No underflow 1= line0 underflowed

21.18.15 DC_WINBUF_T_MEMFETCH_CONTROL_0

Note: MEMFETCH_CLK_GATE_OVERRIDE should only be enabled AFTER the vertical scaler settings in the active copy - VDDA_INCREMENT is properly set up.

Memfetch Control Register

Offset: 0x82b | Byte Offset: 0x20ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	T_MEMFETCH_CLK_GATE_OVERRIDE: Disable clock-gating of memfetch for a controller (all three windows) 0 = DISABLE 1 = ENABLE
0	0x0	T_MEMFETCH_RESET: Reset memfetch for a controller (all three windows) 0 = DISABLE 1 = ENABLE

21.18.16 DC_WINBUF_T_SCRATCH_REGISTER_0_0

Offset: 0x82d │ Byte Offset: 0x20b4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	T_SCRATCH_REGISTER_0: Scratch register 0

21.18.17 DC_WINBUF_T_SCRATCH_REGISTER_1_0

Offset: 0x82e │ Byte Offset: 0x20b8 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

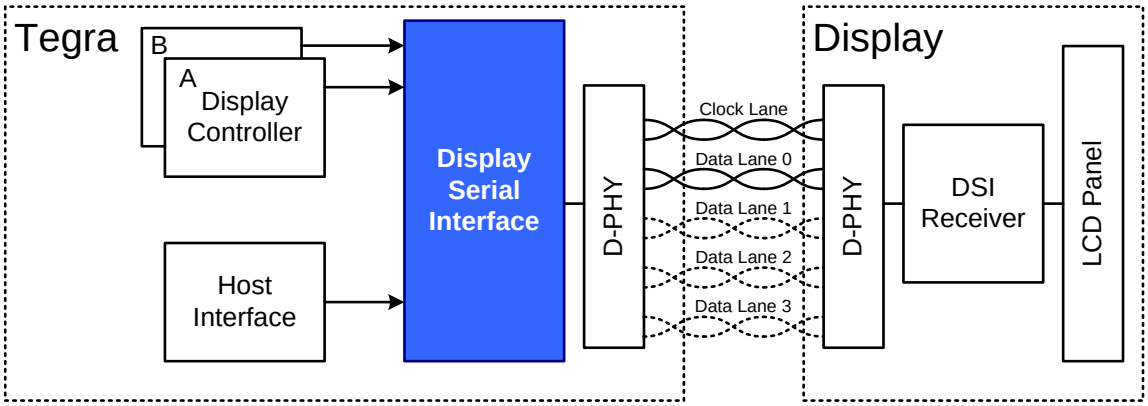
Bit	Reset	Description
31:0	0x0	T_SCRATCH_REGISTER_1: Scratch register 1

[THIS PAGE INTENTIONALLY LEFT BLANK]

22.0 MIPI-DSI (DISPLAY SERIAL INTERFACE)

The Display Serial Interface (DSI) is a MIPI standard serial bitstream that provides a low pin-count interface to a display panel. DSI reduces both package pin-count and I/O power consumption compared with parallel solutions. It transfers pixel data from either one of the display controllers (internal to the Tegra® K1 device) to an external third-party LCD module. The physical positioning of the DSI module in relation to other units/devices in the system is shown in Figure 69.

Figure 69: System Block Diagram of Single DSI Unit



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.

22.1 Features

- Each DSI interface has 1 clock lane and 4 data lanes
- Maximum 1.5 GHz HS transmit rate
- Maximum 10 MHz LP receive rate
- Supports up to 8 PHY lanes with each DSI instance/channel supporting 1, 2, 3, 4 data lanes in Ganged mode configuration
- Video Mode with display controller as master
- Command Mode with host and/or display controller as masters
- Maximum Video Mode resolution is 16:9 AR up to and including (3200x2000), 60 fps at 24 bpp
- Maximum Command Mode packet length is 1920 words

22.2 Functionality

22.2.1 Display Controller Interface

The interface between the display controller and the DSI unit consists of pixel data, Sync data, and a Line-Type code.

DSI captures the state of the line type code every line. This information is then used to look up a pre-programmed packet sequence that corresponds to that line type. There are several different line types to select from.

22.2.2 Packet Poster

The packet poster determines which packets will be sent on what video line when the DSI is in any of the 3 video modes and is being controlled by the display controller. The functions that this sub-unit performs are:

- Line type look-up
- Packet sequence generation
- Pixel data / packet stream integration

There are currently 6 valid line types that can be encoded on the 3 bits from the display controller. The other 2 line types are reserved for future use. The line type coding is shown in the following table.

Table 80: Line Type Descriptions

Line Type	Description
0	Blank line starting with VS
1	Blank line starting with VE
2	Blank line starting with HS
3	Active line
4	First blank line after active
5	First active line
6, 7	Reserved

The packet sequences corresponding to each line type are held in a pair of 32-bit registers. Each register holds three packet descriptors, and each packet descriptor has the following information:

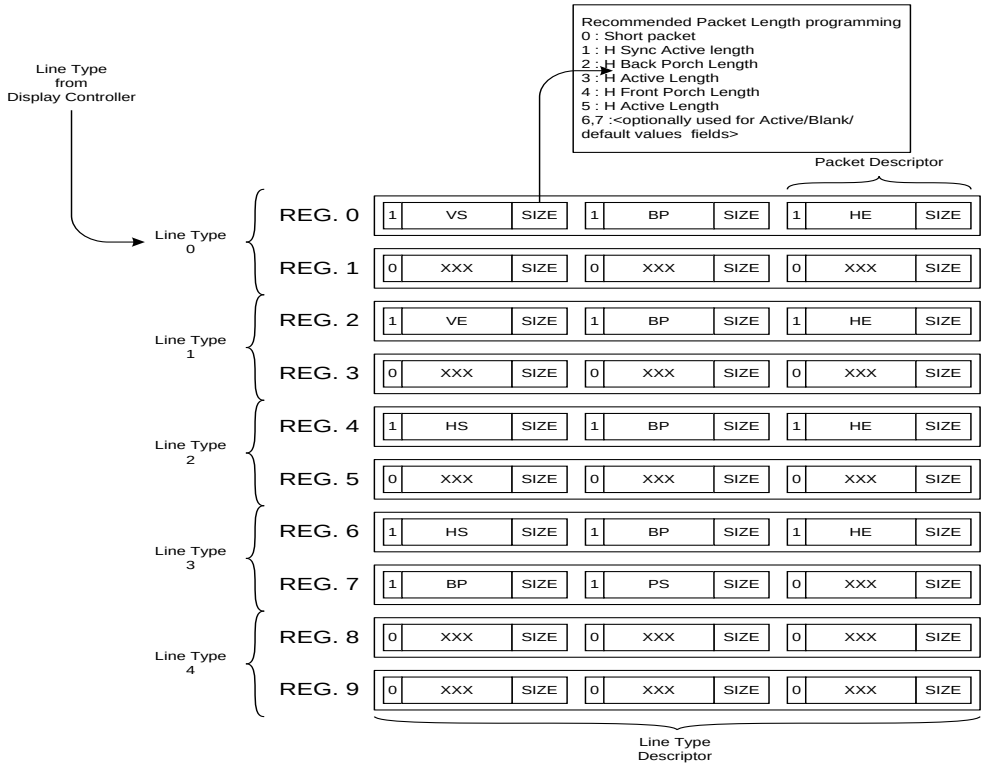
- Valid bit
- 6-bit packet ID
- 3-bit length index

The valid bit simply states whether or not a packet should be generated from that descriptor. A zero valid bit implies that no more packets in this packet sequence should be generated. Thus, the valid bit acts like a null terminator for the list of packet descriptors.

The packet ID is simply the MIPI DSI packet ID value. Software can therefore program any sequence of packets required for that line type.

Rather than have a 16-bit packet length field for every packet descriptor, the length is replaced by a 3-bit index which addresses the full 16-bit length for that packet type. This length is held elsewhere in separate length registers. A complete picture of the packet description data structures is shown in the following figure. This diagram shows typical packet mnemonics for the packets to be generated for video mode with sync pulses. By changing the programming of the packet sequences, any of the video modes, including burst mode, can be realized.

Figure 70: Line Type Look-up and Packet Sequence Descriptors



22.2.3 Host Interface

While many displays use a raster-based packet stream, in some applications it is more efficient to have a frame store in the display device itself and to only update the contents of the frame store when the required pixels have changed. For these types of displays, there is a software-accessible interface for sending command packets using the DSI command mode operation. This mode should also be used to access control and status registers within the display device.

The Host interface consists of a small FIFO for holding packet information and a special register for writing data into the FIFO. In this way, several packets may be queued up by software and sent in one HS burst. This is more efficient that sending the packets one at a time.

When very long sequences of packets are required to be sent – for example, when the host interface is emulating video mode, or where a large amount of data needs to be sent to the display device in one packet – it is possible for the host to use the large video line-store FIFO.

22.2.4 Clocking

The clocks in the DSI include:

- Host Clock = 300 MHz {Maximum frequency}
- Application/Lane Management Byte Clock = function of lanes, pixel clock, pixel depth. 10 to 187.5 Mbps
- Fixed LP receive clock = 20 to 216 MHz
- Lane Bit clock = 40 to 500 MHz differential clock, DDR data (80 to 1500 bps).

More details on pixel clock/byte_clock selection are explained in the next section.

Pixel Clock / DSI Byte Clock Ratios

When running the DSI interface in one of the two non-burst video modes, the relationship between the DSI D-PHY bit-rate clock, the DSI byte-rate lane clock, and the display controller pixel clock must be exact. This ensures 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 81. As can be seen, for some modes, these ratios are not simple powers of 2.

Table 81: Pixel Clock: Byte Clock for Various Modes

Lanes	Pixel Data Format			
	16 bpp	18 bpp (packed)	18 bpp	24 bpp
1	1:2	4:9	1:3	1:3
2	1:1	8:9	2:3	2:3
3	3:2	12:9	1:1	1:1
4	2:1	16:9	4:3	4:3
5	5:2	20:9	5:3	5:3
6	3:1	8:3	2:1	2:1
7	7:2	28:9	7:3	7:3
8	4:1	32:9	8:3	8:3

Because the pixel clock is effectively derived from the D-PHY bit clock, it can be more instructive to look at the number by which you must divide the bit clock to get the correct pixel clock. This information is shown in Table 82. Here, the numbers look reasonable except for the problematic cases of 16bpp over a 3-lane link and 18 bpp (packed) over a 4 lane link. For these two situations, you cannot simply divide the bit rate clock by some integer to arrive at the pixel clock. Moreover, if you need a 50:50 duty cycle for the pixel clock, then you will need to toggle the pixel clock every N/2 bit clock cycles, where N is the number in Table 82. However, if you do this, 18 bpp (packed) over 2 lanes also becomes a problem.

Table 82: 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
5	16/5	18/5	24/5	24/5
6	8/3	3	4	4
7	16/7	18/7	24/7	24/7
8	2	9/4	3	3

Table 83: Value of Shift Clock Divider to Get Pixel Clock (Non-Burst Mode)

Lanes	Data Format (16 bpp)	Data Format (18 bpp)	Data Format (24 bpp)
1	14	16	22
2	6	7	10
3	3.333334	4	6
4	2	2.5	4
5	1.6	1.8	2.4
6	0.666667	1	2
7	1.14285714	1.285714286	1.714285714
8	0	0.25	1

Note: The fractional values in the above table cannot be supported through configuration fields because of a possible limitation in programming the shift divider for deriving the pixel clock from the byte clock in non-burst mode, that is, the SHIFT_CLK_DIVIDER field of the DC_DISP_DISP_CLOCK_CONTROL_0 field. This limitation occurs only in non-burst video mode.

To resolve these issues, derive the display controller pixel clock from a separate PLL and then lock that PLL to the DSI D-PHY PLL using a phase comparator and the appropriate divide ratios obtained from the tables.

DSI PLL – Clock Mux

In the Tegra K1 device, the DSI implements a clock mux for the clock selection of the write clock domain to the line buffer and its upstream path in the DSI.

A single DSI instance receives a pixel data stream from either of the display controller heads. The single bit register field dsi_vid_src in the dsi_clk domain selects the active display head. The selected display pixel clock provides the necessary clocking to the front end of the DSI.

In cases where the Host transmission uses the line buffer, dsi_clk can provide the necessary clocking using the single bit register field pkt_wr_fifo_sel.

22.3 Modes of Operation

The DSI operates in two transmission modes: Video and Host/Command

22.3.1 Video Mode

Communication with the peripheral is by isochronous data transfer similar to a typical video interface. There is no Bus Turn Around permitted in Video Mode, though there is a mandatory period of LP operation at least once per frame. There are three different sub-modes of Video Mode. These are outlined below:

Non-Burst Mode with Start and End

In this mode, the DSI must match the timing of a traditional video raster as closely as possible. This means all information about the start and end of parameters like vertical and horizontal sync, front and back porches must be conveyed to the receiving peripherals via the timing of the transmission of the High-Speed sync packets.

Non-Burst Mode without Start and End

This is like Non-Burst Mode with Start and End, except that the Sync-Active and Sync-End packets are not sent. Pixels must still be delivered at the correct rate.

Burst Mode

Only the timing of Sync-Start packets is required to be accurately conveyed in this mode. The data rate of the pixel data is arbitrary and is typically higher than the pixel rate of the peripheral.

22.3.2 Host/Command Mode

Communication with the peripheral is by asynchronously timed and variable length packets. The packets may contain video data (Display Controller/Host) or control data. In Command Mode, return (read) data can be requested from the peripheral. The DSI will issue a Bus Turn Around (BTA) request and relinquish control of the bus to the peripheral.

22.4 FIFO Buffers

22.4.1 Video Mode Operation

22.4.1.1 Writing Data

The following packets are written into the large data FIFO based on the various triggering events described.

Table 84: FIFO Trigger Events

Trigger Event	Packet
Leading edge of Vertical Sync.	VSync Start
Trailing edge of Vertical Sync	VSync End
Leading edge of Horizontal Sync, *unless* simultaneous with leading or trailing edge of vertical sync	HSync Start
Trailing edge of Horizontal Sync	HSync End
Immediately following VS, HS, VE or HE packets. Packet Word Count determined by display controller timing parameters.	Blanking packet
Immediately following blanking packet	Pixel Stream packet (16, 18, or 24 bits/pixel)

22.4.1.2 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.

22.4.2 Command Mode Operation

22.4.2.1 Writing Data from the Display Controller

Upon receipt of the leading edge of Vertical Sync, the display controller sends whatever DCS command packets are required to initialize the display to receive pixels. Then, as pixels start to arrive from the display controller, a DCS Long Write packet is sent – one per line – which contains the pixel data for that line. No synchronization or blanking packets should be sent. Data packets continue to be sent until the end of the active period. Like video mode packets, there is no need to calculate ECC or CRC words, as this is performed by the hardware on the other side of the data FIFO.

The Packet Sequence registers behave in the same manner as they do for Video Mode packet generation with the following exceptions:

- For Long packets, a DCS command ID byte must be inserted as the first byte of the data payload of the packet, ahead of the pixel data. The packet header Word Count must be 1 more than the number of bytes in the pixel data to

take into account this extra byte. It is the responsibility of software to make sure the packet length is programmed correctly.

- For Line Type 4, any data contained in the Host Data FIFO should be appended to the end of any packets defined in the Packet Sequence for Line Type 4. In this way, the Host may send DCS commands to the Display Peripheral while the Display Controller is sending pixel information to the Display Peripheral.

22.4.2.2 Writing Data from the Host

The host should choose which data FIFO – small or large – to use prior to sending packet information. The large data FIFO is only available if the display controller is not currently using it. The software is responsible for constructing whatever packets are required for the desired operation and should be written into the FIFO via the host. There is no need for the software to compute ECC or CRC words, because this is performed by the hardware on the other side of the data FIFO.

22.4.2.3 Reading Data

The initiation of the reading of data from the FIFO is triggered by a FIFO threshold. Once the threshold is reached, the D-PHY starts to drain the FIFO, and the packets are sent to the display. The threshold should be set such that there is no danger of either FIFO overflow from excessive data from the host or display controller, or FIFO underflow caused by a lack of sufficient data in the FIFO.

22.5 Programming Guidelines

The packet timing diagrams in this section use the following key:

Figure 71: Video Mode Timing Diagram Key

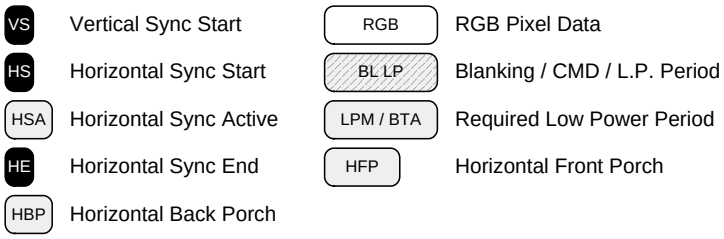


Table 85: Timing Diagram Parameter List

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

22.5.1 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.

22.5.1.1 Video Mode Start-up

All 3 video modes – Burst, Non-Burst and Non-Burst with Sync Ends –have the same start-up sequence with a slight variation between Burst and Non-Burst modes:

- Set up the clocks. This involves configuring and enabling the DSI PLL (PLLD). For Non-Burst and Non-Burst with Sync End modes, the Display Controller must also be programmed to take its pixel clock from the DSI PLL. For Burst Mode, the Display Controller can take its pixel clock either from the DSI PLL or from another clock source. Program the PLLD registers with the correct OSC_FREQ and program the PLLD_BASE register. In addition, the PLLP must be running in order to provide the LP receive clock. Program the PLLP registers with the correct OSC_FREQ and program the PLLP_BASE register.
- Depending on the sub-mode, the various Packet Sequence registers and Packet Length registers must be programmed.
- Set the `VID_TX_TRIG_SRC` field in the `DSI_CONTROL` register to `SOL` to select which display controller will source the video data.
- Enable the DSI
- Program the display controller to produce the raster size required
- Enable the display controller

When the display controller starts sending SOL and data, the DSI automatically starts creating and transmitting packets to the display device.

22.5.1.2 Command Mode Start-up

- Set up the clocks. Program and enable the DSI PLL (PLLD). In addition, the PLLP must be running in order to provide the LP receive clock. Program the PLLP registers with the correct OSC_FREQ and program the PLLP_BASE register. Unlike the Video modes, the selection of the PLLD output clock frequency is essentially arbitrary. The clock frequency should be based on the expected data throughput and the requirements of the particular display device. Program the PLLD registers with the correct OSC_FREQ and program the PLLD_BASE register. In addition, the PLLP must be running in order to provide the LP receive clock. Program the PLLP registers with the correct OSC_FREQ and program the PLLP_BASE register.
- Set the HOST_TX_TRIG_SRC field in the HOST_DSI_CONTROL registers to `IMMEDIATE`. Set any other state required.
- Enable DSI.

The DSI should now be ready to accept command packets written to the DSI_WR_DATA register.

22.5.2 Video Mode Programming

The programming model has been designed to accommodate current video mode sequences and potentially future variation. Each line of video output is associated with a “line type”. This line type is automatically generated by the display controller hardware according to the following table.

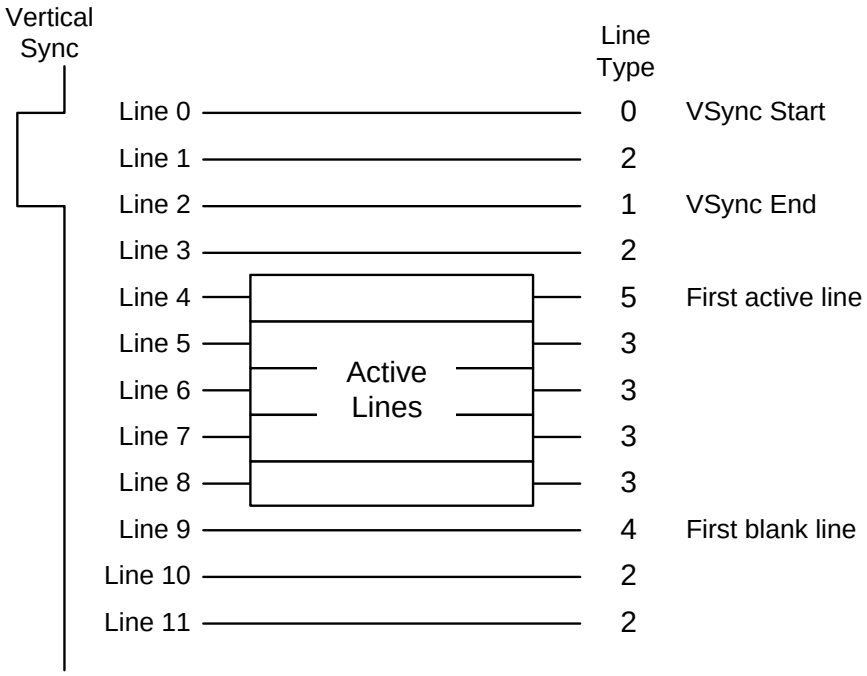
Table 86: Recommended Line Types

Line Type	Description	Typical Raster Position
0	Blank line starting with VS	First line of the raster
1	Blank line starting with VE	Line corresponding to Vsync end.
2	Blank line starting with HS	Vertical blanking line
3	Active line starting with HS	Active line
4	First blank line after active	First blank line after active
5	First active line	First active line

Line Type	Description	Typical Raster Position
6, 7	Reserved	

Figure 72 illustrates a typical raster showing the relationship between various events in the raster and the line types.

Figure 72: 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 87: 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 88 shows how all six packet descriptors fit into the pair of 32-bit packet sequence registers.

Table 88: 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

Register	Field	Bits	Description
	PKT_01_EN	19	Packet 1 Enable
	PKT_02_SIZE	22:20	Packet 2 Size
	PKT_02_ID	28:23	Packet 2 ID
	PKT_02_EN	29	Packet 2 Enable
	SEQ_0_FORCE_LP	30	End in LP state
DSI_PKT_SEQ_0_HI	PKT_03_SIZE	2:0	Packet 3 Size
	PKT_03_ID	8:3	Packet 3 ID
	PKT_03_EN	9	Packet 3 Enable
	PKT_04_SIZE	12:10	Packet 4 Size
	PKT_04_ID	18:13	Packet 4 ID
	PKT_04_EN	19	Packet 4 Enable
	PKT_05_SIZE	22:20	Packet 5 Size
	PKT_05_ID	28:23	Packet 5 ID
	PKT_05_EN	29	Packet 5 Enable

Finally, the SIZE field in the packet descriptor is used as a pointer to a 16-bit "*packet length*" field in the packet length registers. An indirect pointer is used rather than storing the packet length directly in the packet descriptor because the packet lengths are physically large – 16 bits – but there is a lot of re-use. There are only a few different lengths used in a typical raster layout. Thus, it is more efficient to hold the lengths in separate length registers and then to simply refer to them with a short pointer. This allows the storing of 36 packet descriptors in only 6 pairs of packet sequence registers.

Note that one of the packet length registers is reserved for short packets. If a SIZE field in the packet descriptor is programmed to 0, this packet is likely a short packet. Short packets are fixed in length to 4 bytes including the packet header byte and ECC byte. In the context of video mode, they are essentially reserved for timing packets. All other packet length registers can be used to determine the length of any associated long packet.

The SEQ_0_FORCE_LP bit is used to determine if the link should be placed in the LP state at the end of the packet transmission. In Burst Mode operation, the link always drops back into LP state at the end of the HS packet transmission on a line. However, for Non-Burst Modes, the HS transmission may continue to the next line. It is important the that the hardware state machine that generates packets not attempt to go to the LP state, but should instead prepare for the next sequence of packets associated with the next video line and keep the line in the HS transmission state.

The packet length registers are shown in Table 89. The packet size pointer in the packet descriptor can point to any of the available length registers – with the one exception of short packets, whose SIZE field must point to length register 0. It is recommended that the suggested packet length assignment for various length registers is followed to reduce confusion when debugging packet descriptors and packet sequences.

Table 89: Packet Length Registers and Assignments

Register	Field	Suggested Assignment
DSI_PKT_LEN_0_1	LENGTH_0	Must be used for short packets
	LENGTH_1	Horizontal sync active length (HSA)
DSI_PKT_LEN_2_3	LENGTH_2	Horizontal back porch length (HBP)
	LENGTH_3	Horizontal active length (ACTIVE)
DSI_PKT_LEN_4_5	LENGTH_4	Horizontal front porch length (HFP)
	LENGTH_5	–Horizontal active length (ACTIVE)
DSI_PKT_LEN_6_7	LENGTH_6/7	<EOTP/other packet IDs>

When programming the length registers, remember that these contain byte counts, not pixel counts. Therefore, the DSI pixel format, number of DSI lanes in use and the finite (non-zero) size of the packet header and CRC words should be taken into

account when programming these values – especially for blanking packets since all of these parameters affect the number of bytes that should be used to emulate a specific blanking time. The equations required to determine the byte count in the packets are given in the examples that follow. The identification of the length given in the tables and equations has been included in three video mode examples so that the values in the examples can be easily tied to the registers.

22.5.3 SOL Delay Programming

In order to ensure that the pixel FIFO from display to DSI does not underflow when in video mode, it is necessary to delay the start of packet generation by the DSI by a fixed amount from the arrival of the SOL signal from the display. This is especially true of Burst-Mode, because the DSI byte clock can be very much faster than display pixel clock. If no delay is applied, the DSI will very quickly consume all the pixels in the FIFO and the FIFO will underflow. The SOL_DELAY registers are used to set this delay.

22.5.3.1 Non-Burst Mode

In non-burst mode, the rate at which the DSI consumes pixels is the same as the rate at which the display module produces them, so it is merely sufficient to ensure that enough pixels have been fed into the FIFO to overcome the internal latency. This internal latency is approximately 8 pixel clock cycles. However, SOL_DELAY is programmed in DSI byte clocks, so the pixel format and the number of lanes being used should be taken into account when programming this value. Contact your NVIDIA FAE for details on the relationship between display pixel clock and byte clock. These ratios are then used to determine the value of SOL_DELAY as follows:

SOL_DELAY = (((Sol2VldDly) + 12) * F_{DSI} / Fpixel) + FifoLatency

FifoLatency = Ceil (2 * (F_{DSI} / Fpixel)) + 6

Sol2VldDly = Hsync start to Pixel valid delay

Where F_{DSI} and F_{pixel} are the DSI byte and pixel clocks, respectively.

Alternatively, a fixed value of SOL_DELAY that will work for all pixel formats and numbers of lanes can be programming by taking the worst-case ratio of 1:3 (pixel:byte) clock and multiplying by 8 to give SOL_DELAY = 24.

22.5.3.2 Burst Mode

In Burst mode, not only must the pixel format, number of lanes, and the DSI / pixel clock ratio be taken into account, but also the horizontal back porch time (including Hsync) and the Horizontal active time. So, for Burst mode:

- When EOTp packets are enabled:

  SOL_DELAY = (((Sol2VldDly + 4 + Tactive + 1) * FDSI / Fpixel) –

  ((Tactive + 1) * FDSI_NB / Fpixel_NB))) + FifoLatency

- When EOTp packets are not enabled:

  SOL_DELAY = (((Sol2VldDly + 4 + Tactive) * FDSI / Fpixel) –

  ((Tactive) * FDSI_NB / Fpixel_NB))) + FifoLatency

That is,

FifoLatency = Ceil (2 * (FDSI / Fpixel)) + 6

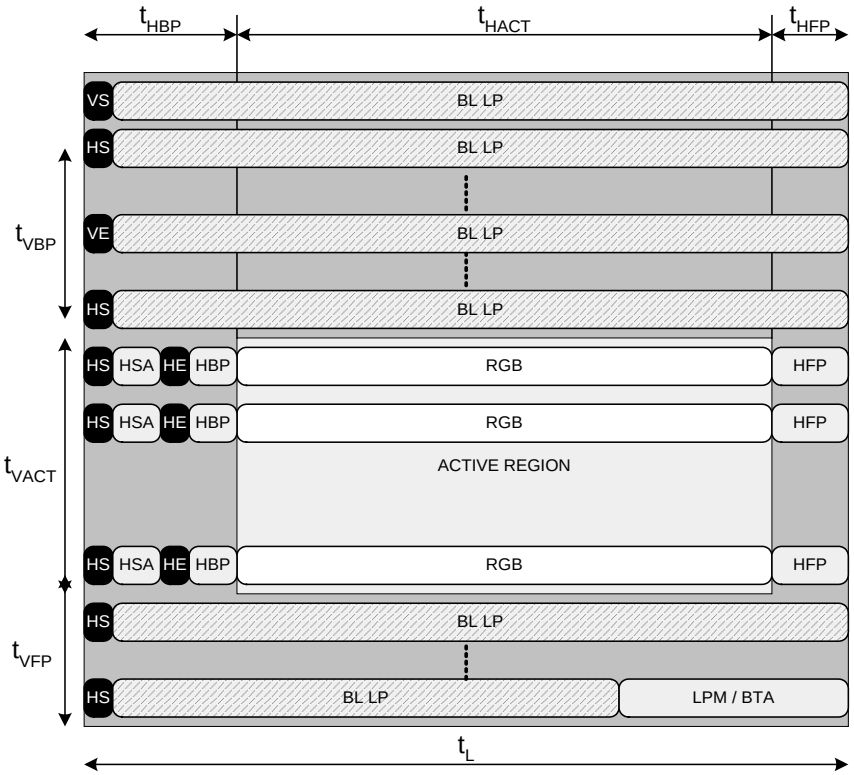
Sol2VldDly = Hsync start to Pixel valid delay

Where the ratio F_{DSI_NB} / F_{pixel_NB} is determined by the clock ratio tables for Non-Burst mode, according to the pixel format used and the number of active lanes.

22.5.4 Non-Burst Mode with Start and End

This mode conveys traditional raster synchronization information across the link to the peripheral by sending both start and end sync packets.

Figure 73: Timing Diagram for Video Non-Burst Mode with Start and End



During the Vertical Active period, all packets should be concatenated into a single HS transmission for the duration of the Vertical Active period.

The figure below shows how a single line is constructed from multiple packets.

Figure 74: Non-Burst Mode with Start and End Packet Timing Detail

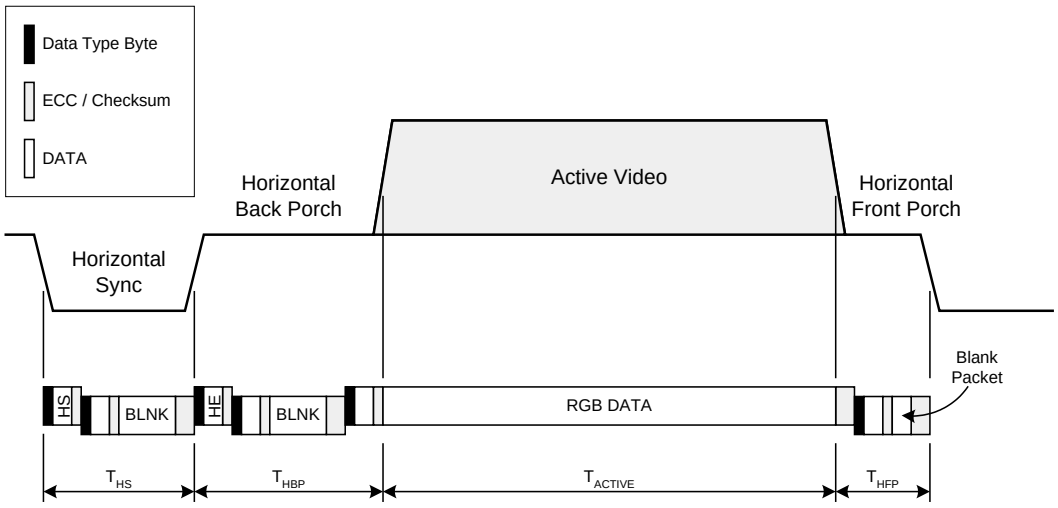


Table 90: Payload Size Table - Non-Burst Mode with Start and End

Packet	Payload Size (Bytes)
HSA	(T_{HS} * B) – 10
HBP	(T_{HBP} * B) – 14
ACTIVE	(T_{ACTIVE} * B)
HFP	(T_{HFP} * B) – 8

B = 2, 2.25 or 3, depending on pixel format.

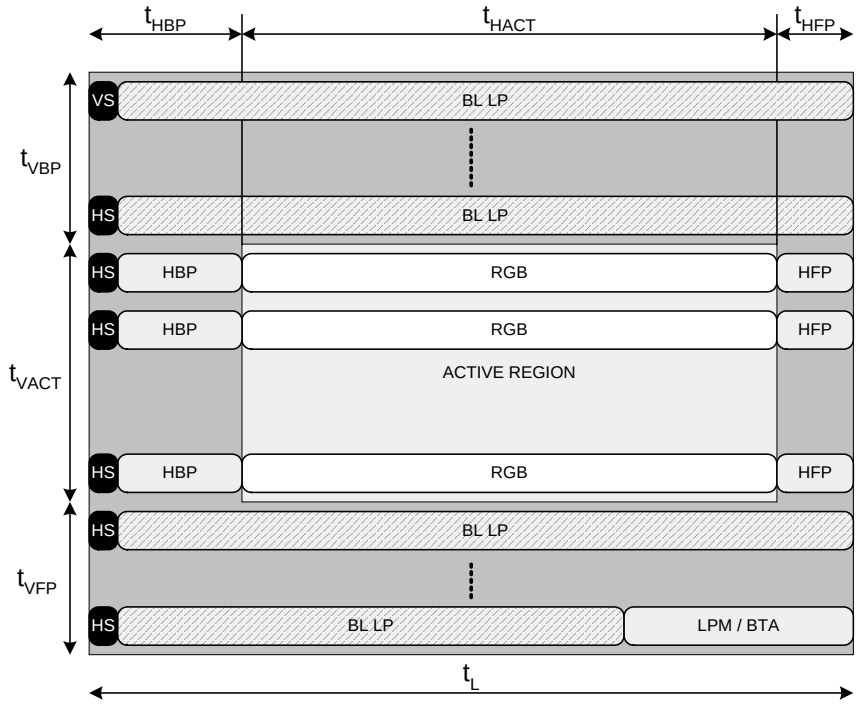
Table 91: Line Type Packet Sequences - Non-Burst with Sync Ends

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0	EOT	7								
1	VE	0	EOT	7								
2	HS	0	EOT	7								
3	HS	0	BLNK	1	HE	0	BLNK	2	RGB	3	BLNK	4
4	HS	0	EOT	7								
5	HS	0	BLNK	1	HE	0	BLNK	2	RGB	3	BLNK	4

22.5.5 Non-Burst Mode (without Ends)

This mode relaxes the requirement to mimic the generation of sync pulses and merely mandates that the start of the line is indicated and that the pixels appear at the same rate and in the same area of the raster as a tradition raster structure.

Figure 75: Timing Diagram for Video Non-Burst Mode



During the Vertical Active period, all packets should be concatenated into a single HS transmission for the duration of the Vertical Active period.

The figure below shows how a single line is constructed from multiple packets.

Figure 76: Non-Burst Mode Packet Timing Detail

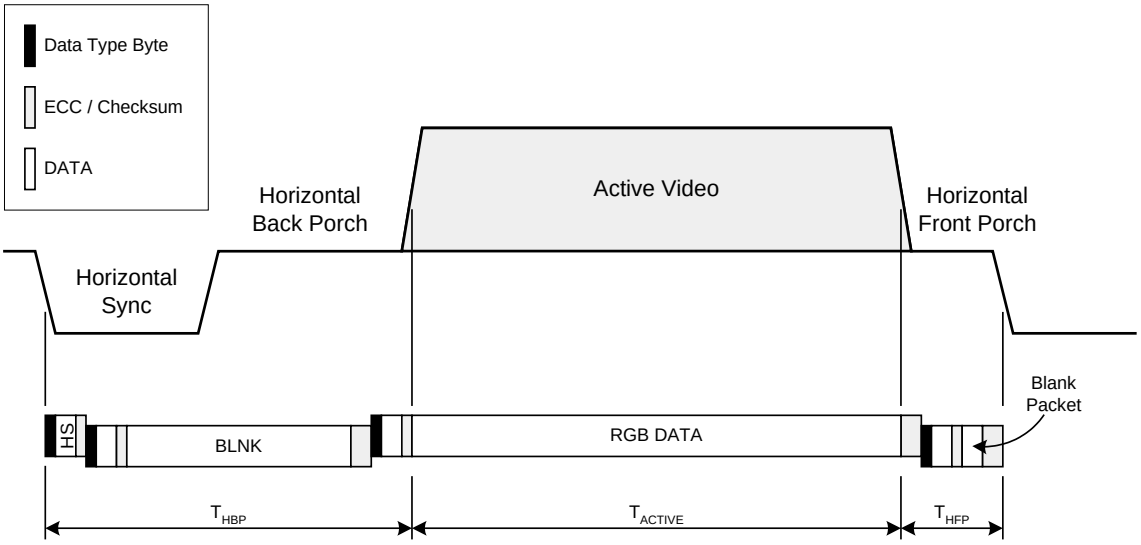


Table 92: Payload Size Table - Non-Burst Mode

Packet	Payload Size (Bytes)
HBP	(T_{HBP} * B) – 14
ACTIVE	(T_{ACTIVE} * B)
HFP	(T_{HFP} * B) – 8

B = 2, 2.25, or 3, depending on pixel format.

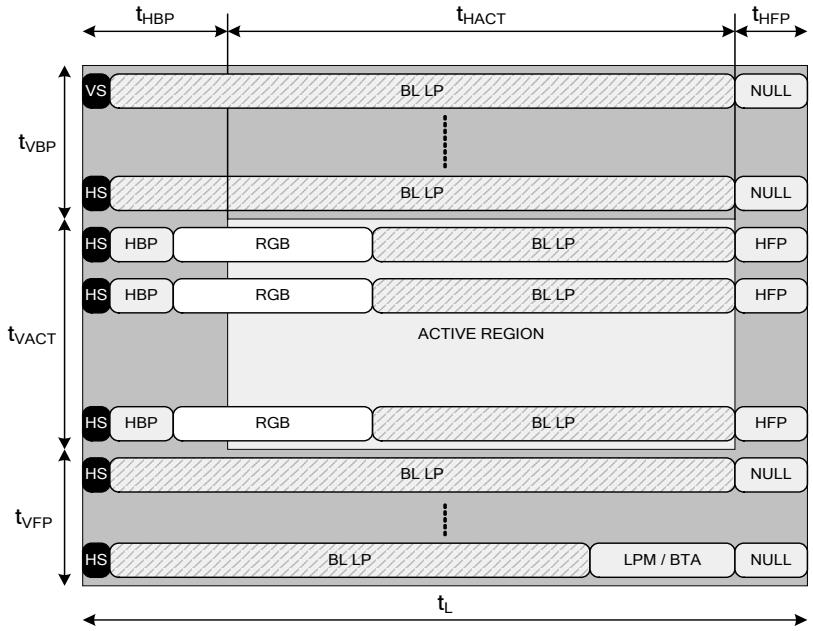
Table 93: Line Type Packet Sequences – Non Burst

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0	EOT	7								
1	HS	0	EOT	7								
2	HS	0	EOT	7								
3	HS	0	BLNK	2	RGB	3	BLNK	4				
4	HS	0	EOT	7								
5	HS	0	BLNK	2	RGB	3	BLNK	4				

22.5.6 Burst Mode

In Burst Mode, the only attempt to match the raster structure is with the timing of the sync start events. The actual RGB pixel data is transmitted at whatever rate is convenient. This means that the HS, HBP, and RGB packets become compressed with respect to the timing of the underlying raster. This allows some period of idle time for each line that can be used for the transmission of other packets.

Figure 77: Timing Diagram for Video Burst Mode



During the Vertical Active period, packets on an individual line should be concatenated into a single HS transmission. However, unlike Non-Burst Mode, the bus should go idle – if possible – at the end of this transmission. This will allow additional non-video packets to be sent simultaneously with the video stream. The NULL and HFP packets may be optional. Check the latest MIPI DSI specification for clarification.

Table 94: Line Type Packet Sequences - Burst Mode

LT	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0	EOT	7				
1	HS	0	EOT	7				
2	HS	0	EOT	7				
3	HS	0	BLNK	2	RGB	3	EOT	7
4	HS	0	EOT	7				
5	HS	0	BLNK	2	RGB	3	EOT	7

Notes:

- Burst mode always forces the LP packet field to LP mode. DSI_DSI_PKT_SEQ_0_LO/HI_0/1/2/3/4/5[SEQ_0_FORCE_LP] = 1’b1
- The register programming updates in video mode that impact the raster timing (DSI_DSI_PKT_SEQ_*_LO/HI_*, video mode control fields DSI_DSI_CONTROL_0) are assumed to be static with respect to the display controller. On-the-fly updates are not supported.

22.5.7 Sequence while Switching the Modes/Updates to Raster Timing

This programming sequence must be followed when switching the modes/updates to raster timing:

1. Disable the DC2DSI interface path (DC_DISP_DISP_WIN_OPTIONS [DSI_ENABLE] = 1’b0)
2. Disable the DSI (DSI_LEG_EN disable)
3. Update the DSI configuration registers.
4. Enable the DC2DSI interface path (DC_DISP_DISP_WIN_OPTIONS [DSI_ENABLE] = 1’b1).

5. Enable the DSI (DSI_LEG_EN enable)

22.5.8 Command Mode Programming

There are two sources of command mode packet sequences:

- Display Controller
- Host Interface

Only one of these sources will be active at any particular time.

22.5.8.1 Command Mode from Host

In this mode of operation, all packets sent over the DSI interface are determined by software. There may be hardware assistance in the generation of error correction codes (ECCs) and cyclic redundancy checks (CRCs), but all other data is passed unaltered to the DSI physical layer.

Host Packet Writes

Packets are written to the hardware by writing to the DSI_WR_DATA register. The data written is passed into the DSI Host transmit FIFO. If the data is a packet header and the ECC_ENABLE field in the HOST_DSI_CONTROL register is set to ENABLE, then the MSBs of the 32-bit packet header word are replaced with a hardware-computed ECC byte prior to being written into the FIFO. If this field is set to DISABLE, then no action is taken by the hardware and the packet header is written unchanged.

If the packet is a long packet, then the packet payload should be written to the register after the packet header is written. If the CS_ENABLE field of the HOST_DSI_CONTROL register is set to ENABLE, then the hardware computes the check-sum and appends it to the packet information. If this field is set to DISABLE, then software must append the correctly computed check-sum to the packet data.

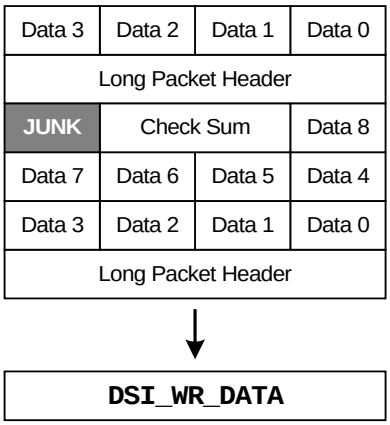
Rules:

1. Software should make sure packets are transmitted in their entirety and that once transmission starts, the FIFO contains enough data to finish a transition on a packet boundary. This can be achieved by only initiating a transmission – either explicitly or indirectly – once all data required for a transmission has been written to the transmit FIFO.
2. In Type-1 Display modules (i.e., the display device includes the full frame buffer to hold image data), the complete image data can be written to the frame buffer via the Host command mode. Software configures the register field to enable the Frame buffer selection for transfer of high-resolution image data.
   DSI_HOST_DSI_CONTROL[PKT_WR_FIFO_SEL]

   Note: The maximum length of the long packet supported is 1920 words including header, payload, and CRC.

3. Packets should be written such that the packet header is always written in one 32-bit word and is never split across writes. In other words, if the end of a packet does not fall on a 32-bit word boundary (if the payload has an odd length, for example), then the header for the next packet should not border the last packet, but should realign with the register. See Figure 78 for an example.

Figure 78: Packet Alignment for Writes to DSI_WR_DATA



Note: For unaligned word transfers, the host transactions are split into multiple host transfers at the unaligned boundary.

Host Packet Transmission

The source that controls when the Host write FIFO starts to drain (flush) is programmable. Selectable source is:

- Explicit FLUSH bit in a control register

Immediate (HOST_TX_TRIG_SRC == IMMEDIATE**)**

As the name implies – if a '1' is written to the `DSI_HOST_TRIGGER` field of the `DSI_TRIGGER` register, then transmission of the data in the Host write FIFO will start immediately. It is recommended that all data required for transmission is written to the DSI_WR_DATA register prior to setting this bit.

22.5.8.2 Command Mode from Display Controller

This mode of operation allows the display controller to write pixel data packets to a DBI-like device that does not require data to be sent in an isochronous raster structure like a DPI device. The data coming from the display controller is sent in an isochronous manner, but the commands sent will be DCS control commands, rather than Video Mode timing and blanking packets.

Setup

There are six steps to operating in this mode:

1. Program the Peripheral display device using DCS commands via the Host Command interface. It is important to send the set_column_address, set_page_address, and set_pixel_format commands. These effectively define the area to which you will be writing pixels.
2. Program the DSI_INIT_SEQ_CONTROL and DSI_INIT_SEQ_DATA_* registers in the DSI interface with the appropriate values. Any commands required to be sent once per frame and *every* frame by way of setup prior to the pixel data being sent should be put in here.
3. Program the `DSI_DATA_FORMAT` field of the DSI_CONTROL register to the required pixel format. Note that `BIT18NP` should not be programmed – see the section below on pixel format restrictions.
4. Program the DSI_PKT_SEQ_* registers. Program the registers in the following way:
   - Packet Sequence registers for Line Types 0, 1, 2, and 4 should all be programmed with 0 in all the Packet Enable fields. In other words, there should not be any packets generated for these line types.
   - Packet Sequence registers for Line Types 3 and 5 should be programmed with a DCS Long Write packet.

5. Program the DCS command ID register to have a write_start command associated with Line Type 5 (First line of active) and a write_continue command associated with Line Type 3 (all other active lines).
6. Enable the display. When Vertical syncs arrive from the display, the initialization sequence should be sent, and for every active line, the pixel data will be sent in a DCS Long Write packet.

Pixel Format Restrictions

The MIPI DCS specification allows for up to 6 different pixel data formats to be transmitted in a write_start or write_continue command. These pixel formats are listed in Table 95. The DSI supports 3 of these pixel formats.

The formats supported are also shown in the “supported” column of Table 95. Do not set `BIT18NP` as a pixel format. There is no DCS equivalent of this format, so undefined behavior may result.

Table 95: DCS Pixel Format Support

DCS ID	DCS Format	Supported	Name
0	reserved	N/A	-
1	3bpp	N	-
2	8bpp	N	-
3	12bpp	N	-
4	reserved	N/A	-
5	16bpp	Y	BIT16P
6	18bpp	Y	BIT18P
7	24bpp	Y	BIT24P

Simultaneous Host Command Packets

It is necessary, from time to time, to send a DCS command to the display peripheral in a “side-band” fashion while the display controller is continuing to send DCS write commands with pixel information. Line Type 4 (first blank line) is reserved to indicate to the hardware when it should attempt to send any DCS command packets requested by software.

If it is desired to send a DCS command in this way, the DCS command packet should be written in its entirety (header, ECC, DCS command, payload, CSC) into the Host Command FIFO. The HOST_TX_TRIG_SRC field should then be set to `IMMEDIATE`.

The DSI hardware will then send the entire contents of the FIFO out on the DSI interface on the first line of blanking. Sending the data on this line guarantees the Host packet transmission will have enough time to complete before the next packet generated by the Display Controller is generated.

> **Note:** There is no BTA permitted in this mode, so no ACK, ACK with error report, or read return data will be generated.

If the host transfer is enabled during the frame video blanking interval, the following sequence needs to be performed at the end of the frame:

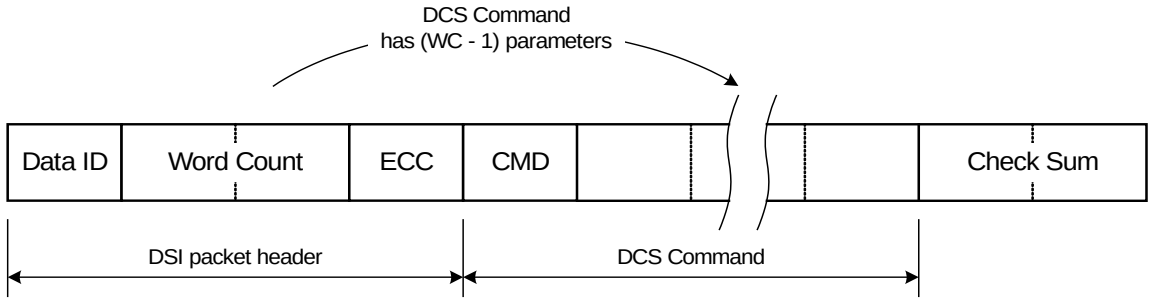
1. Wait for an end of frame event from the display controller.
2. Set the increment syncpt soft reset; that is, program the INCR_SYNCPT_SOFT_RESET field in the DSI_INCR_SYNCPT_CNTRL_0 register to 1’b1.
3. Clear the increment syncpt soft reset; that is, program the INCR_SYNCPT_SOFT_RESET field in the DSI_INCR_SYNCPT_CNTRL_0 register to 1’b0.
4. Continue with the subsequent configuration.

22.5.9 Display Command Set (DCS) Packets

DCS is a legacy control command set that is used to program various control registers present in typical LCD panels used in cell phones. Historically, the commands would be sent to the display over a MIPI DBI interface. Since MIPI DSI is meant to replace both DPI and DBI, it must also transport these control commands.

There are several DCS-specific commands in the MIPI DSI specification that can be used to send DCS commands to the Peripheral Display Device. However, the most useful is probably the DCS Long Write packet since it can be used to send commands with more than one parameter. The DCS command is embedded with an MIPI DSI Long Packet as follows: The DCS Command Byte and all the DCS Command Parameters (payload) are concatenated and sent in the Payload of the MIPI DSI Long Packet. The Word Count of the DSI packet conveys the total size of the DSC packet. As usual, a 2-byte CS footer is appended to the DSI Long Packet. The next figure below shows how a DCS Long Write packet is constructed..

Figure 79: DCS Command Placement in DSI Long Packet



16 bpp Byte Ordering in 16 bpp Format for Video/Command Mode

The DSI specification suggests a different byte order in the 16 bpp video and command mode data bit order, i.e., data packing in:

Video mode [16:0] -> {B[4-0],G[5-3]},{G[2-0],R[4-0]} (Default)

Command mode [16:0] -> {G[2-0],B [4-0]},{R[4-0],G[5-3]} (SwapEn)

The byte order can be configured through the DSI_DSI_CONTROL_0 register, DFMT_16BPP_SWAP_EN field. By default, the video mode order is selected and the command mode can be selected by the programing DFMT_16BPP_SWAP_EN to 1.

Frame Synchronization Between Identical Command Mode Displays

When stretching a single surface across multiple identical displays, frame synchronization can be achieved to avoid visual artifacts due to timing drifts across both the displays.

DCS command “adjust_vsync_timing” provides a mechanism for adjusting the command mode display vsync/frame sync timing with respect to another identical display. Vsync can be shifted, i.e., delayed/advanced by the specified number of scan lines.

Host Command Mode Byte Transmission

Command mode operation primarily involves sending the commands and data to the peripheral (the display device). The host processor indirectly controls activity at the peripheral by sending commands, parameters, and data to the display controller. Command Mode operation requires a bidirectional interface capabilities.

Command mode transfers are classified into,

- HS/LP mode

  Host command mode transmission can be enabled in High Speed and Low Power modes. Follow this programming sequence to configure the DSI to enable command mode transactions.

- Configure DSI_DSI_CONTROL_0 [DSI_HOST_ENABLE]
- LP/HS mode of transmission is selected by configuring the DSI_HIGH_SPEED_TRANS in the DSI_HOST_DSI_CONTROL_0 register.
- Enable checksum/Ecc mode via DSI_HOST_DSI_CONTROL_0[CS_ENABLE], …, DSI_HOST_DSI_CONTROL_0[ECC_ENABLE].
- Write the host data to the register DSI_DSI_WR_DATA_0. Once all the command packets are pushed into the FIFO, trigger the start of the transaction via the HOST_TX_TRIG_SRC field in the DSI_HOST_DSI_CONTROL_0 register.

- Raw Mode

  Host command mode transmission also supports RAW data bytes to be transferred to the display device. In Raw mode, the data written to the host is transferred with no concept of packets and the DSI hardware does not decode the packet headers and ECC/CS computation is not performed.

  In order to send the RAW data bytes (debug / diagnostic workaround mode):

  - Program the number of bytes to be transferred to the DSI_RAW_DATA_BYTE_COUNT register. The byte count programmed is equal to the total number of raw bytes (not to be interpreted as the byte count of a packet).
  - Write raw data bytes to the DSI Host.
  - Enable RAW_DATA in the DSI_HOST_DSI_CONTROL_0 register.

22.5.10 High-Speed Clock Configurations

In High Speed mode, the Host provides a low-swing, differential DDR clock signal for high-speed data transfers. The high-speed clock lane is configurable via the DSI_HS_CLK_CTRL] field in the DSI_DSI_CONTROL register.

- Free Running Mode /Continuous

  The D-Phy clock lane operates in HS free running or continuous mode; i.e., used in systems where the D-Phy clock lane acts as a clock source to the display devices eliminating the need for an alternate oscillator clock source.

- TX Mode/Discontinuous Mode

  The D-Phy clock lane operates in burst mode or discontinuous mode; i.e., the D-Phy clock lane remains active only during the HS transmission and stops the clock lane when there are no high-speed transmissions active in any of the data lanes.

Notes:

1. The Host programs the total horizontal blank period to meet the following criteria to provide enough time for HS <-> LP transitions.

   (HBLANK) >= (number_of_lanes * (HS_Trail + HS-Exit + CLK_POST + CLK_TRAIL + HS_EXIT + TLPX + TLPX + TCLK_PREPARE + TCLK_ZERO + TCLK_PRE + 1 + TLPX + THS_PREPARE + TDAT_ZERO + 1))

   The parameters in this equation are defined in the DSI_DSI_PHY_TIMING registers.

2. The DSI_HS_CLK_CTRL field in the DSI_DSI_CONTROL_0 register control field for the HS clock lane is programmed when the DSI_HIGH_SPEED_TRANS control field in the DSI_DSI_CONTROL_0 register is programmed for HS packet transmission of packets.
3. The HS clock lane state transitions from continuous to discontinuous with soft reset assertion.

22.5.11 Ultra-Low Power Sequence

In this procedure, starting from the Stop state, the transmit side drives the TX-ULPS-Rqst state (LP-10) and then drives the TX-ULPS State (LP-00). After this, the clock lane enters the Ultra-Low Power state. If an error occurs, and an LP-01 or LP-11 is detected immediately after the TX-ULPS-Rqst state, the Ultra-Low Power state entry procedure is aborted, and the receive side waits for or returns to the Stop state, respectively.

The Clock lane and Data lanes can be configured to enter and exit ULPS independently/simultaneously:

- Independent control of the ULPS

  The Clock needs to be in the HS state, sending the active clock when data lanes are entering ULPS, When entering ULPS, the data lanes enter the ULPS first, followed by the clock lanes. Similarly, when exiting ULPS, the Data lanes exit the ULPS first, followed by the clock lanes.

  DSI_ULTRA_LOW_POWER_CONTROL register configures the control to enter the ULPS on particular clock/Data lane.

  ULPS Entry

  - Program the data lane to enter ULP sequence, wait for syncpt.
  - Program the clock lane to enter ULP sequence and, wait for syncpt

  ULPS Exit

  - Program the data lanes to exit the sequence, wait for syncpt
  - Program the clock lanes to exit the sequence, wait for syncpt
- Simultaneous control of the ULPS

  The DSI_ULTRA_LOW_POWER field in the HOST_DSI_CONTROL register controls the ULPS for all lanes (clock and data lanes 0-3) simultaneously.

Notes:

- The DSI_TWAKEUP field in the DSI_PHY_TIMING_2 register defines the length of the exit interval in multiples of 512 byte clocks common for all lanes (Clock/Data).
- The ULPM entry sequence always starts from the NORMAL state. Software should place the control fields in the NORMAL state after the exit sequence after the last operation is done. That is, in case of a ULPM sequence on clock, data lanes, you are recommended to program the register to the NORMAL state after the data lane exit sequence.

22.5.12 End of Transmission Packet (EOTp)

The DSI specification defines a dedicated End of Transmission packet (EoTp) at the protocol layer to indicate the end of HS transmission. For backwards compatibility with earlier DSI systems, this EoTp can be enabled or disabled.

For host transactions during the video mode, EOTp is programmed in the Packet Sequence register, and host data is sent on the interface followed by the last video packet byte, including the EOTp packet as the last word. That is, software programs the EOTP packet as part of the host data during host transactions during video mode.

22.5.13 Host Command Packet During Video Mode Transmission

DSI supports transmission of host command packets during vertical blanking in Video mode.

Host packet transmission during vertical blank time is possible and programmable through the DSI_VID_MODE_CONTROL register to enable and select the line type to trigger the host command packets during video mode. The 0/1/2/4 line types with single short packet transmission as part of video raster can enable simultaneous host transfer in Video mode.

Video blank line types are:

- Line Type = 0 - Blank line starting with VS
- Line Type = 1 - Blank line starting with VE
- Line Type = 2 - Blank line starting with HS
- Line Type = 4 - First Blank line after active starting with HS.

The programming sequence to enable host transactions during Video mode is:

1. Choose the line type 0/1/2/4 (Video Blanking) for simultaneous host transfers during Video mode. Program the DSI_LINE_TYPE field in the DSI_DSI_VID_MODE_CONTROL_0 register to FOUR.
2. Simultaneous host transfers are enabled during Video mode via the Video mode control field. Program the DSI_CMD_PKT_VID_ENABLE field in the DSI_DSI_VID_MODE_CONTROL_0 to 1'b1.
3. Write host data transactions to the host data buffer via the DSI_DSI_WR_DATA_0 register.

HS Packet	Host Data Packets (<= 64 Words)

4. Trigger host transmission when the complete host data for transmission is pushed into the buffer. Program the DSI_HOST_TRIGGER field in the DSI_DSI_TRIGGER_0 to 1'b1.
5. Wait for FRAME_DONE syncpt from the display controller.
6. Set the increment syncpt soft reset; that is, program the INCR_SYNCPT_SOFT_RESET field in the DSI_INCR_SYNCPT_CNTRL_0 register to 1'b1.
7. Clear the increment syncpt soft reset; that is, program the INCR_SYNCPT_SOFT_RESET field in the DSI_INCR_SYNCPT_CNTRL_0 register to 1'b0.

Notes:

- The host packet data transmission cannot be enabled in Video burst mode, where NULL_PACKETs are included with the HSYNC packet.
- Checksum/ECC computation for the host packets can be performed by either the hardware or software by programming the CS/ECC_Enable fields in the DSI_HOST_DSI_CONTROL_0 register.
- After a single VSYNC/VSYNCEND/HSYNC short packet is transmitted, if the Host transmission is enabled, software-constructed Host data packets are transmitted.
- The length of the Host FIFO data is limited to 64 words deep.
- Host data should be pushed into the Host FIFO before the frame is triggered.
- Host packets must be aligned to a word boundary. For host packets ending on an unaligned word boundary, there is support for a maximum of 4 such consecutive packets transferring in a worst-case configuration scenario and relaxed for other configurations. In the worst-case configuration scenario, the number of lanes is 4, and the Host packets end on Byte 0 (that is, a single byte is written in the last word).

  For an unaligned word boundary, if the end of a packet does not fall on a 32-bit word boundary, then the header for the next packet should not border the last packet (see the figure showing packet alignment for writes to DSI_WR_DATA).

22.5.14 Ganged Mode (Odd-Even/Left-Right) Programming

The Tegra K1 DSI supports Ganged mode. This section assumes that the reader is familiar with the DSI specification and normal video mode programing guidelines.

Odd-Even

For Odd-Even Ganged mode, the programming guidelines are based on these assumptions/limitations:

1. Both partitions have a single PLLD and clock source (DSI instances)

2. For VNB, all data lanes end in the same clock cycle aligned to the number of configured lanes.
3. The number of pixels in the 18 bpp packed case is always a multiple of four.
4. The number of active pixels from the display controller is not greater than 4096 pixels.

Programming guidelines/sequence and equations for Odd-Even Ganged mode are listed below:

1. Required inputs are: image resolution, data format, and optional number of lanes
2. Compute the pixel clock frequency from the given resolution :

   pixel frequency = (TotalHorizontalPeriod * TotalVerticalPeriod * FrameRate) / 10^6
3. Determine the shift clock divider value and ganged mode imagesplit information from the number of pixels, data format, and pixel frequency:

   pixel frequency = DSI clock (byte frequency) * 4/clock divider
4. Symmetrical split is possible for all resolutions. Determine the possibility of an asymmetrical split.

   Let n1 = number of lanes for the first partition, and n2 = number of lanes for the second partition from step 3.

   totalNumLanesInGangedMode = n1 + n2
5. Compute the correction pixels needed to align the totalNumPixels, totalNumBytes with totalNumLanesInGangedMode. From Limitation #2 and #3:
   - The pixelsNeededToInitiallyAlign should be such that updated totalNumPixels is a multiple of LCM (total Num Lane sInGangedMode,X)
   - totalNumBytes = totalNumPixels*bytesPerPixel is a multiple of totalNumLanesInGangedMode.

   Note:
   - LCM – least common multiple (X=4 for 18bpp packed, otherwise X=1)

     If remainder is non-zero/ganged alignment and was not successful (from Step 9), add ‘n’ number of additional active pixels (zeros/random data) so that the updated image active width is divisible by the LCM above. This is needed to support image split as determined in (Step 4).
   - Add the additional correction/dummy pixels to HFP, determine the new pixel frequency, and compute the byte frequency using the clock divider. Use this information to program the PLLD and shift clock divider. It is okay to add dummy active pixels since panels provide an option using a register field that can be configured to drop/discard the padded pixels.
   - Update the total number of pixels now with the newly added active pixels to ensure the total number of active pixels does not exceed 4096 pixels. In case the total number of pixels exceeds 4096 pixels, then the frame is considered to not support asymmetrically split for the given configuration (i.e., frame rate and split configurations).
   - In the 18 bpp packed format case, you cannot achieve alignment over a definite set of iterations for a given frame rate. If the alignment is not achieved within 50-60 pixels, the pixel alignment is assumed to not be possible for the frame for symmetric split at that frame rate.
6. Determine if the (first partition width) : (second partition width) ratio = n1/n2. If successful, proceed to the next step. If not, ganged width alignment is not successful. Repeat from step 5 by adding more active pixels that still meet the requirement mentioned there.

   splitFactor (for first partition)= n1/ totalNumLanesInGangedMode;

   First Partition Width (active pixels)= CEIL(spliFactor*image width)

   Pixels per line of first Partition = CEIL(splitFactor*totalNumPixels)

   second partition active width = imagewidth – first partition width

   pixels per line of second partition = totalNumPixels - Pixels per line of first Partition

Note:

- In consideration of constraint #4, software will ensure that the total number of pixels does not exceed 4096 pixels. If the total number of pixels exceeds 2570 pixels, then the frame does not support asymmetrically split for the given configuration, i.e., frame rate and split configurations.
- The number of active pixels added is limited to no more than 60 pixels. If this value exceeds 60 in these iterations, the input image cannot be split as per the split ratio (cannot deviate much from the selected frame rate).

7. Determine the new pixel frequency (new active pixels and HFP correction) and compute the byte frequency using the clock divider. Use this information to configure the clock parameters for PLLD clock generation and the shift clock divider.
8. Determine the factors of the first partition width and second partition width.

   Select ‘x’ from the factors of the first partition and ‘y’ from the factors of the second partition such that x:y = n1/n2. The minimum of 1:1 (symmetrical) is possible for any image.

   - The first group of pixels: ganged start pointer = 0, valid width = x and low width = y
   - The second group: ganged start pointer = x, valid width = y, low width = x
9. In Ganged mode, the packet sequence programming includes HSYNC, HACT, and HFP (total Horizontal blanking period) packets.

   HACT payload size = Total Actual Image Bytes on Data lanes

   Initial HFP payload size = totalNumBytesForTheSelectedPartition - numActBytesToBeSent - 16 (16 is because of 4 Bytes - HSYNC, (4+2) Bytes - HACT, (4+2) Bytes - HFP)
10. Compute the correction bytes (per line of partition) to align with the line width and number of lanes (per partition), limitation #2

    correctionBytesPerLane = CEIL(totalNoOfPixelsPerHorzLine *(FDSI/ Fpixel) – (BPP/ Gangedmode_lanes))

    totalNumBytesForTheSelectedPartition = (correctionBytesPerLane * numOfLanesForSelectedPartition)

    total Horizontal Blank(Bytes) = totalNumBytesForTheSelectedPartition - 16 + correctionBytesForAlignment
11. Ganged mode register programming
    - Even Partition Programming

      DSI_DSI_GANGED_MODE_CONTROL_0[DSI_GANGED_MODE_EN] = 1;

      DSI_DSI_GANGED_MODE_START_0[DSI_GANGED_START_POINTER] = 0

      DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_HIGH_WIDTH] = Even partition active width

      DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_LOW_WIDTH] = Even partition inactive/low width.
    - Odd Partition Programming

      DSI_DSI_GANGED_MODE_CONTROL_0[DSI_GANGED_MODE_EN] = 1;

      DSI_DSI_GANGED_MODE_START_0[DSI_GANGED_START_POINTER] = High width of Even partition

      DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_HIGH_WIDTH] = odd partition active width.

      DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_LOW_WIDTH] = odd partition inactive/low width
12. SOL delay calculation:

    SolFactor = ((Sol2VldDly + 6) * (FDSI / Fpixe)) + 6;

    sol_delay = Ceil (TotalHorzPixelWidth * (FDSI / Fpixel)) -

Ceil (((SplitFactor * TotalHorzPixelWidth) * BPP) / Gangedmode_lanes) + SolFactor;

Final_sol_delay = (dcs_mode) ? (sol_delay + 20) : sol_delay

DSI_DSI_SOL_DELAY_0 = Final_sol_delay

Note:

- Sol2VldDly = Hsync start to Pixel valid delay
- totalNoOfPixelsPerHorzLine = Total Number of horizontal pixel count.
- SplitFactor = DSI_partition_lanes / Gangedmode_lanes.
- Gangedmode_lanes = Total number of lanes required in Ganged mode transfer
- DSI_partition_lanes = Total lanes in current partition/channel.
- BPP = Bytes per pixel

Left-Right:

For Left-Right Ganged mode, the programming guidelines are based on these assumptions/limitations:

- Both partitions have a single PLLD and clock source (DSI instances)
- For VNB, all data lanes end in the same clock cycle aligned to the number of configured lanes (per DSI specification).
- The number of pixels in the 18 bpp packed case is always aligned to 4 pixels (per DSI specification).
- The number of active pixels from the display controller is not greater than 4096 pixels.

Programming guidelines/sequence and equations for Left-Right Ganged mode are listed below:

1. Required inputs are Image resolution,data format, and optional number of lanes
2. Compute the pixel clock frequency from the given resolution
   pixel frequency = (TotalHorizontalPeriod * TotalVerticalPeriod * FrameRate) / 10^6
3. Determine the possible shiftclock divider value and ganged mode image split the information from the number of pixels, data format, and pixel frequency.

   pixel frequency = DSI clock (byte frequency * 4/clock divider)
4. Symmetrical split is possible for all resolutions. Determine the possibility of the asymmetrical split.
5. Let n1=number of lanes for first partition and n2 = number of lanes for second partition from step 3

   Total number of DSI data lanes required for video refresh:
   totalNumLanesInGangedMode = n1 + n2
6. Compute the correction/additional pixels needed to align the total number of pixels and total number of bytes with totalNumLanesInGangedMode.

   From Limitation #3:

   - The pixelsNeededToInitiallyAlign should be such that an updated totalNumPixels is a multiple of LCM (totalNumLanesInGangedMode,X)
   - totalNumBytes = totalNumPixels*bytesPerPixel
     Ensure the total number of bytes is a multiple of totalNumLanesInGangedMode

   Note: For the least common multiple (LCM), X = 4 for 18 bpp packed; otherwise X = 1

   In the 18 bpp packed format, in case the alignment does not complete over a definite set of iterations for a given frame rate, check that the total additional pixels added to the align is not greater than 50-60 pixels. If this range is exceeded, the pixel alignment is not possible, and the frame is not considered to be asymmetrically split for the 18 bpp packed case (for that frame rate).

Add the additional correction pixels to the HFP, determine a new pixel frequency, and compute the byte frequency using the clock divider. Use this information to configure the clock parameters for PLLD clock generation and the shift clock divider.

7. Determine the number of pixels per split on updatedPixelCount from step 6:
   - o Determine the first partition width, and then the remaining pixels will be in the second partition.

     splitFactor (for first partition)= n1/ totalNumLanesInGangedMode

     First Partition Width (active pixels)= CEIL(splitFactor*image width)

     Pixels per line of the first partition = CEIL(splitFactor*totalNumPixels)

     second partition active width = imagewidth – first partition width

     pixels per line of the second partition = totalNumPixels - Pixels per line of the first partition.

     Note: For 18 bpp packed, software should recheck if each of the widths determined above is multiple of 4. Otherwise, adjust the active pixels of the partition (increase) to align to 4 pixels and correspondingly decrease the pixels in the HFP (to still meet the line time) accordingly. Adjust pixels in the second partition as well.

     numActBytesToBeSent = partition active width * bytesPerPixel

     totalNumBytesForTheSelectedPartition = pixels per line of selected partition * bytesPerPixel

8. In Ganged mode, the packet sequence programming includes HSYNC, HACT, and HFP (total horizontal blanking period) packets.

   HACT payload size = Total Actual Image Bytes on Data lanes

   Initial HFP payload size = totalNumBytesForTheSelectedPartition - numActBytesToBeSent - 16

   (16 is because of 4 bytes - HSYNC, (4+2) Bytes - HACT, (4+2) Bytes - HFP)

9. Compute the correction bytes (per line of partition) to align with line width and number of lanes (per partition) (limitation #2)

   Correction Bytes added per lane (correctionBytesPerLane) = CEIL(totalNoOfPixelsPerHorzLine *(FDSI/ Fpixel) – (BPP/ Gangedmode_lanes))

   Correction Bytes added per partition (totalNumBytesForTheSelectedPartition) = correctionBytesPerLane * numOfLanesForSelectedPartition

   total Horizontal Blank(Bytes) = totalNumBytesForTheSelectedPartition – 16 + correctionBytesForAlignment

10. Ganged mode register programming
    - o Left Partition Programming

      DSI_DSI_GANGED_MODE_CONTROL_0[DSI_GANGED_MODE_EN] = 1;

      DSI_DSI_GANGED_MODE_START_0[DSI_GANGED_START_POINTER] = 0;

      DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_HIGH_WIDTH] = Total horizontal active width

      DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_LOW_WIDTH] = Total horizontal width – Total horizontal active width.
    - o Right Partition Programming

      DSI_DSI_GANGED_MODE_CONTROL_0[DSI_GANGED_MODE_EN] = 1;

      DSI_DSI_GANGED_MODE_START_0[DSI_GANGED_START_POINTER] = leftPartitionWidth (for right partition)

      DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_HIGH_WIDTH] = Total horizontal active width

DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_LOW_WIDTH] = Total horizontal width – Total horizontal active width.

11. SOL delay:

    SolFactor = ((Sol2VldDly + 6) * (FDSI / Fpixe)) + 6;

    sol_delay = Ceil (TotalHorzPixelWidth * (FDSI / Fpixel)) -

    Ceil (((SplitFactor * TotalHorzPixelWidth) * BPP) / Gangedmode_lanes) + SolFactor;

    Final_sol_delay = (dcs_mode) ? (sol_delay + 20) : sol_delay

    DSI_DSI_SOL_DELAY_0 = Final_sol_delay

Note:

- Sol2VldDly = Hsync start to Pixel valid delay
- totalNoOfPixelsPerHorzLine = Total Number of horizontal pixel count.
- SplitFactor = DSI_partition_lanes / Gangedmode_lanes.
- Gangedmode_lanes = Total number of lanes required in Ganged mode transfer
- DSI_partition_lanes = Total lanes in current partition/channel.
- BPP = Bytes per pixel

22.5.14.1 Recommended Packet Sequence

Packet	Payload Size (Bytes)
ACTIVE	(T_{ACTIVE} * B)
HFP	(T_{HFP} * B) – 8

- B = 2, 2.25 or 3, depending on pixel format.
- N = Number of Lanes used.

Table 96: Recommended Payload Size - Ganged Mode

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0								
1	HS	0								
2	HS	0								
3	HS	0	RGB	3	BLNK	4				
4	HS	0								
5	HS	0	RGB	3	BLNK	4				

22.5.14.2 Alternate Packet Sequence

Packet	Payload Size (Bytes)
ACTIVE	(T_{ACTIVE} * B)
HFP	(T_{HFP} * B) – 8

- B = 2, 2.25 or 3, depending on pixel format.

- N = Number of Lanes used.

Table 97: Optional Payload Size- Ganged Mode

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0								
1	HS	0								
2	HS	0								
3	HS	0	BLNK	2	RGB	3	BLNK	4		
4	HS	0								
5	HS	0	BLNK	2	RGB	3	BLNK	4		

22.5.15 Soft Reset Programming

Soft reset is asserted in the DSI through the LEG_DSI_ENABLE control field of the DSI_DSI_POWER_CONTROL_0 register. When soft reset is asserted, the DSI controller transitions the internal states to default mode. When it is asserted between valid transactions, the current transaction is halted, the internal states are moved to default mode, and a new transaction starts by programming the desired values following the programming sequence explained in the other sections.

Note: For Host transmissions enabled through the DSI_HOST_TRIGGER field of the DSI_DSI_TRIGGER_0 register, soft reset cannot be issued between a transaction, and software will write 1'b0 to the register to clear the trigger bit.

22.5.16 Function Programming

22.5.16.1 ECC Generation

For precise details on the calculation of the ECC field of the packet header, reference should be made to the MIPI Alliance Standard for Display Serial Interface. 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.

22.5.16.2 CRC Insertion

The DSI checksum used for long packets is derived using the generator polynomial $x^{16}+x^{12}+x^5+x^0$. Details of this can be found in section 9.6 of the MIPI DSI specification. To understand how the CRC is generated, think of the packet data as consisting of a continuous serialized stream of bits, rather than a sequence of 8-bit bytes. When thought of in this way, it is relatively straight forward to generate the CRC. Reproduced below is an abridged version of the example C code contained in Appendix B of the MIPI DSI specification.

```
// Polynomial, bit reversed form (since DSI transmits LSB first) …
const unsigned short CRC16GenerationCode = 0x8408;

unsigned short CalculateCRC16( unsigned char *DataStream_ptr, unsigned short NumberOfDataBytes)

{
  unsigned short ByteCounter;
  unsigned char  BitCounter;
  unsigned char  CurrentData;
  unsigned short CRC16Result = 0xFFFF;

  if (NumberOfDataBytes > 0) {
    for (ByteCounter = 0; ByteCounter < NumberOfDataBytes; ByteCounteer++) {
      CurrentData = DataStream_ptr[ByteCounter];
      for (BitCounter = 0; BitCounter < 8; BitCounter++) {
        if ((CRC16Result & 0x0001) ^ (CurrentData ^ 0x0001))
          CRC16Result = ((CRC16Result >> 1) & 0x7FFF) ^ CRCGenerationCode;
        else
          CRC16Result =  (CRC16Result >> 1) & 0x7FFF;
        CurrentData = (CurrentData >> 1) & 0x7F;
      }
    }
  }
  return CRC16Result;
}
```

This code may not be very high-performance due to the inner for loop which iterates over bits, rather than bytes. While it is instructive and could form the basis of a reference piece of code, it is not recommended where performance is important. There are many documented methods of performing this calculation in parallel in order to speed up the computations. These methods are beyond the scope of this document.

22.5.17 Read Data Return

DSI is a bidirectional interface. Data is returned from the peripheral display device only after the display controller has requested information by issuing a Bus Turn Around (BTA). All returned data is written into a FIFO that can be read using Host reads of a DSI register.

22.5.17.1 Reading Peripheral Registers

A typical application of a BTA is in the reading of a register from the display peripheral. This is achieved in the following way:

1. Set up the DSI interface to be in Host-driven command mode.

2. Set the DSI_MAX_THRESHOLD to 3.
3. Set the HOST_TX_TRIG_SRC field of the HOST_DSI_CONTROL register to FIFO_LEVEL.
4. Set the PKT_BTA field of the HOST_DSI_CONTROL register to ENABLE.
5. Write a DCS READ command packet (see section 8.8.8.2 of the MIPI DSI specification) into the Command FIFO by writing to the DSI_WR_DATA register.

This will result in the transmission of a DCS READ packet to the peripheral, the initiation of a BTA and – assuming the peripheral received the packet without error – the return of the requested data to the Host Read Return FIFO. The data can then be read from the FIFO by reading the DSI_RD_DATA register.

22.5.17.2 Bus Turn Around

- BTA is only supported for Host driven Command Mode interface. There will be no BTA during video mode transmission.
- Whether or not a BTA is initiated is controlled by the PKT_BTA field of the HOST_DSI_CONTROL register.
- The DSI Read Return FIFO is 4 bytes wide and 8 entries deep, so 32 bytes in total. this is enough to hold 8 short packets, or a mixture of short and long packets. The length of long packets must be severely restricted.
- Hardware does not perform ECC or CS checks on the read return data. Entire packets are simply made available to software to perform whatever checks they desire.
- There is the ability to increment a sync point counter on the arrival of the returned data or on the receipt of an error report in the event there was a problem with the read packet transaction.
- Software will ensure to program the byte clock frequency dsi_clk greater than 52 MHz, when performing BTA transactions.

22.5.17.3 BTA – Response Time Parameter

When a peripheral receives a READ Request, it is expected that a Bus Turn Around will immediately follow to extend support for the display device that cannot handle the request immediately after the Read request is issued.

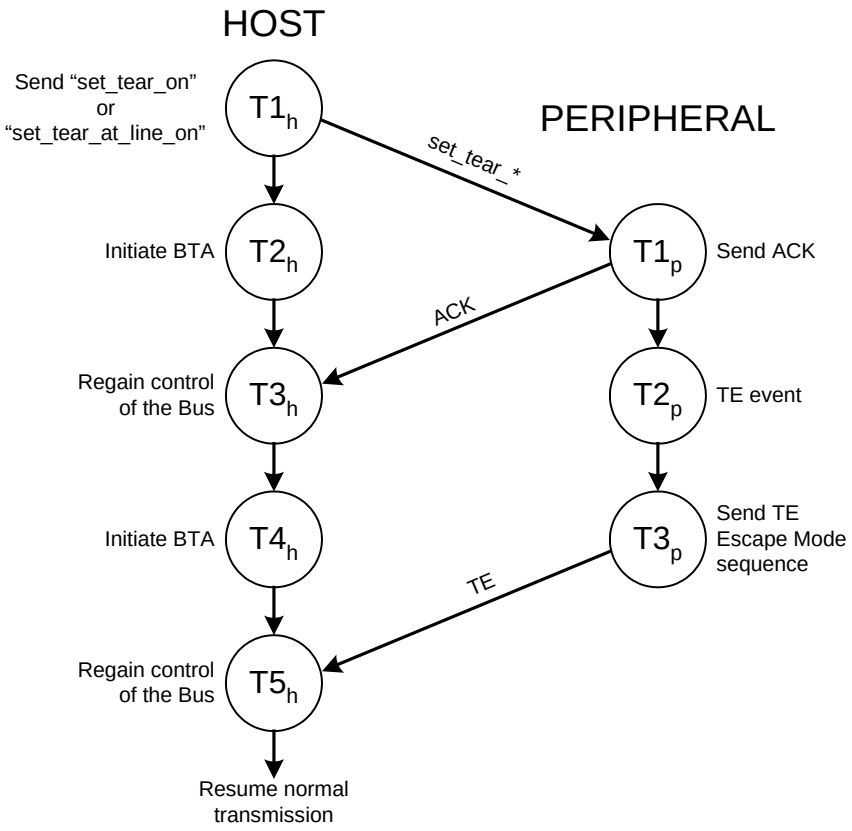
The DSI_DSI_BTA_TIMING_0 register defines DSI_TPKTBTA to allow a programmable delay between the end of host transmission and generation of BTA request for extending support for the display devices with slow response.

Note: Valid programmable values for all the DSI_DSI_BTA_TIMING_0 register fields (DSI_TTAGO/DSI_TTASURE/DSI_TTAGET/DSI_TPKTBTA) are 0 to 254. Invalid value is 255.

22.5.18 Tearing Effect

In order to synchronize the update of a Command Mode display with data from the Host, a signal from the display can be sent to indicate when it is safe to proceed with the transmission of new data. This is the Tearing Effect reporting signal. Refer to section 8.12 “TE Signaling in DSI” of the DSI specification for more details.

Figure 80: Tearing Effect State Transition Diagram



Programmatically, this is achieved in the following way:

1. Send SET_TEAR_ON or SET_TEAR_AT_LINE_ON command with the PKT_BTA field in the HOST_DSI_CONTROL register set to ENABLE. This is state T1h.
2. Wait for ACK to come back from the peripheral. This is states T2h and T3h.
3. Set the IMM_BTA field in the HOST_DSI_CONTROL register set to ENABLE. This will cause the D-PHY to go into BTA without having to send a command first. This is state T4h.
4. Wait for TE return byte from the peripheral. This is state T5h.

22.5.19 Pad Calibration Programming

- Configure the MIPI calibration clock to operate at 72 MHz via programming the register CLK_RST_CONTROLLER_PLLP_BASE_0 register.
- Enable the mipi calibration clock via Set CLK_ENB_MIPI_CAL to 1.
- Release reset to mipi_cal logic by deasserting SWR_MIPI_CAL_RST to 0
- Configure the DSI pad/Bias pads to appropriate values

  DSI_PAD_CONTROL_3_0= 0x0

DSI_PAD_CONTROL_4_0 = 0x0

MIPI_CAL_MIPI_BIAS_PAD_CFG0_0 = 0x0;

MIPI_CAL_MIPI_BIAS_PAD_CFG1_0 = 0x00020000

MIPI_CAL_MIPI_BIAS_PAD_CFG2_0 = 0x00000000

- Configure MIPI calibration settings for DSI pads for setup MIPI_CAL_DSI* _MIPI_CAL_CONFIG [*SELDSI*/ *OVERIDEDSI*/ *HSPDOSDSI*/ *HSPUOSDSI*/ *TERMOSDSI*].

  MIPI_CAL_DSI*_MIPI_CAL_CONFIG_0

- Enable all the DSI lanes that require calibration driving LP_11 state.
- MIPI calibration start via - MIPI_CAL_MIPI_CAL_CTRL_0[MIPI_CAL_STARTCAL]
- Monitor the status via MIPI_CAL_CIL_MIPI_CAL_STATUS_0[*].

22.5.20 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.

22.5.20.1 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 98.

Table 98: Peripheral Response to Various Conditions

Condition	Non-Read Packet	Read Packet
No error	ACK	Read Data
Corrected single bit error	ACK with ERR	Read Data + ACK with ERR
Non-corrected multi-bit error	ACK with ERR	ACK with ERR
SOT, EOT, VI ID or other D-PHY error	ACK with ERR	ACK with ERR

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 single byte trigger has the value 0x84. Since the return FIFO is 32 bits wide, the 0x84 value will be placed in the bottom 8 bits of the FIFO.

Read Data

If a read command packet is sent, then BTA request should be enabled to allow the peripheral to return the read data. If no error occurs in the transmission of the read packet, or a corrected single bit error occurs, then the read data will be returned to the host DSI hardware in the form of a read packet. The precise form will depend on the type of read packet transmitted. Refer to the MIPI DSI specification, section 8.10 for details.

If a corrected error occurs during the transmission of the read command, the requested read data will be returned in the normal way, but an ACK with Error report packet will be appended to the read return data packet.

ACK with Error Report

The error report is contained in the 16 payload bits of a special short packet returned by the peripheral. The bits allocations of the packet data are detailed in section 8.9.5 of the MIPI DSI specification but are repeated here for convenience.

Table 99: 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

22.5.21 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 100: 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 100, but will also cause a tally counter to increment so that software can read the register and determine if any of the time outs have occurred. Software will be able to reset these tally counters to 0 by writing to the tally register. The value written will be irrelevant. The PR_TO will report its operation in a different manner. When the PR_TO counter is actually counting, there will be a status bit that reads as '0'. When the PR_TO terminal count is reached and time out ends, the status bit will become '1'. This will allow software to determine when the peripheral has been reset.

Table 101: 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

22.6 MIPI-DSI Registers

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

22.6.1 DSI_INCR_SYNCPT_0

Offset: 0x0 | Byte Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = COND_4 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15 16 = COND_16 17 = COND_17 18 = COND_18 19 = COND_19 20 = COND_20 21 = COND_21 22 = COND_22 23 = COND_23 24 = COND_24

Bit	Reset	Description
		25 = COND_25 26 = COND_26 27 = COND_27 28 = COND_28 29 = COND_29 30 = COND_30 31 = COND_31
7:0	0x0	INDX: syncpt index value

22.6.2 DSI_INCR_SYNCPT_CNTRL_0

Offset: 0x1 | Byte Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal states of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all affected Host1x clients, then clear all SOFT_RESETs.

22.6.3 DSI_INCR_SYNCPT_ERROR_0

Offset: 0x2 | Byte Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

22.6.4 DSI_CTXSW_0

Context switch register. Should be common to all modules. Includes the current channel/class (which is writable by software) and the next channel/class (which the hardware sets when it receives a context switch).

Any context switch request triggers an interrupt to the host and causes the new channel/class to be stored in NEXT_CHANNEL/NEXT_CLASS (see vmod/chexample). Software sees that there is a context switch interrupt and does the necessary operations to make the module ready to receive traffic from the new context. It clears the context switch interrupt and writes CURR_CHANNEL/CLASS to the same value as NEXT_CHANNEL/CLASS, which causes a context switch acknowledge packet to be sent to the host. This completes the context switch and allows the host to continue sending data to the module.

Context switches can also be pre-loaded. If CURR_CLASS/CHANNEL are written and updated to the next CLASS/CHANNEL before the context switch request occurs, an acknowledge will be generated by the module, and no interrupt will be triggered. This is one way for software to avoid dealing with context switch interrupts.

Another way to avoid context switch interrupts is to set the AUTO_ACK bit. This bit tells the module to automatically acknowledge any incoming context switch requests without triggering an interrupt. CURR_* and NEXT_* will be updated by the module so they will always be current.

Offset: 0x8 | Byte Offset: 0x20 | Read/Write: R/W | Reset: 0xf000f800 (0bxxxxxxxxxxxxxxxx11111x0000000000)

Bit	R/W	Reset	Description
31:28	RO	X	NEXT_CHANNEL: Next requested channel
25:16	RO	X	NEXT_CLASS: Next requested class

Bit	R/W	Reset	Description
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.

22.6.5 DSI_DSI_RD_DATA_0

DSI Read Return Data

Offset: 0x9 │ Byte Offset: 0x24 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_RD_DATA: Each read to this register will pop 32 bits from the 32-bit wide read return data FIFO. The FIFO has NV_DSI_HOST_DATA_RETURN_FIFO_DEPTH entries.

22.6.6 DSI_DSI_WR_DATA_0

Host FIFO Write Input

Offset: 0xa │ Byte Offset: 0x28 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_WR_DATA: Each write to this register will push 32 bits into the 32-bit wide Host data FIFO. FIFO has NV_DSI_HOST_DATA_FIFO_DEPTH entries

22.6.7 DSI_DSI_POWER_CONTROL_0

Display Power Control

DSI Enable

Offset: 0xb │ Byte Offset: 0x2c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	LEG_DSI_ENABLE: DSI interface Enable 0 = DISABLE : Disable DSI 1 = ENABLE : Enable DSI

22.6.8 DSI_INT_ENABLE_0

Interrupt Enable Register

Offset: 0xc │ Byte Offset: 0x30 │ Read/Write: R/W │ Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	CTXSW_INT_ENABLE: Context Switch Interrupt Enable 0 = DISABLE : interrupt disabled 1 = ENABLE : interrupt enabled

22.6.9 DSI_INT_STATUS_0

Interrupt Status Register

Offset: 0xd │ Byte Offset: 0x34 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CTXSW_INT: Context switch interrupt status (clear on write)

22.6.10 DSI_INT_MASK_0

Interrupt Mask

Setting bits in this register mask the corresponding interrupt but does not clear a pending interrupt and does not prevent a pending interrupt to be generated. Masking an interrupt also does not clear a pending interrupt status and does not prevent a pending interrupt status to be generated.

Offset: 0xe │ Byte Offset: 0x38 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CTXSW_INT_MASK: Context Switch Interrupt Mask 0 = MASKED : interrupt masked 1 = NOTMASKED : interrupt not masked

22.6.11 DSI_HOST_DSI_CONTROL_0

DSI Control Register When Input is from HOST

Offset: 0xf │ Byte Offset: 0x3c │ Read/Write: R/W │ Reset: 0x00000043 (0bxxxxxxxxxx00x000xx00xx0001000011)

Bit	Reset	Description
21	0x0	FIFO_STAT_RESET: write only bit to clear FIFO underflow/overflow flags. If a new underflow/overflow event occurs during the same time, current access cannot clear the status bits
20	0x0	CRC_RESET: Write only bit. When written with a 1, causes the Verification CRC generator to reset to 0xFFFF_FFFF. If written with 0, it has no effect.
18:16	0x0	DSI_PHY_CLK_DIV: Phy clock divider value for byte clock 0 = DIV1 1 = DIV2
13:12	0x0	HOST_TX_TRIG_SRC: Controls the source of the trigger to start sending packets. 0 = SOL : Start of Line signal from the Display Controller. 1 = FIFO_LEVEL : How full the FIFO is. Level determined elsewhere. 2 = IMMEDIATE : Determined by a write to the DSI_HOST_TRIGGER field of the DSI_TRIGGER register
9:8	0x0	DSI_ULTRA_LOW_POWER: Ultra Low Power 0 = NORMAL 1 = ENTER_ULPM 2 = EXIT_ULPM
7	0x0	PERIPH_RESET: Initiate an Escape Mode Peripheral Reset. 0 = DISABLE : Causes an Escape Mode Command to be sent to reset the 1 = ENABLE : External display device. Also starts the PR_TO counter. PR_TO state can

Bit	Reset	Description
		be checked with the P_RESET_STATUS field in the DSI_TO_TALLY register. Hardware clears this bit upon completion of issuing "Trigger Reset" OR called "Reset Entry" command is sent out.
6	0x1	RAW_DATA: Host raw data mode. In this mode. All data is sent exactly as written. No attempt to decode packet headers is made. This bit will also override the function of the ECC and CS ENABLE fields. No ECC or CS will be generated. This mode is intended as a debug / diagnostic workaround mode only. 0 = DISABLE : Normal mode. 1 = ENABLE : Enable raw data transmission.
5	0x0	DSI_HIGH_SPEED_TRANS: DSI high speed transmission of packets 0 = LOW : Low speed - Note: Unlikely ever to be used. 1 = HIGH : High speed
4	0x0	PKT_WR_FIFO_SEL: Host Write FIFO Select. In video mode, software shall not program PKT_WR_FIFO_SEL=VIDEO. 0 = HOST : Write data to the small host data FIFO only. 1 = VIDEO : Write data to both the host and video line store FIFO, in series. Note: Software shall only program PKT_WR_FIFO_SEL field in Host mode.
3	0x0	IMM_BTA: Generate BTA immediately, e.g., for Tearing Effect reporting. Note: This will generate a BTA and pass control of the D-PHY to the remote peripheral without the need to send any packet. Once the BTA is initiated on interface this bit gets cleared. Later on Host syncpt opdone is returned when the remote peripheral has responded and relinquished control of the bus. 0 = DISABLE : Do not generate BTA 1 = ENABLE : Generate BTA immediately, without waiting for a packet.
2	0x0	PKT_BTA: Generate BTA at the end of Host packets 0 = DISABLE : Do not generate BTA 1 = ENABLE : Generate BTA after the next packet is sent.
1	0x1	CS_ENABLE: enable Hardware Check Sum (CS) for Host packets Note: when CS is disabled, Host is responsible for generating proper CRC and adding the 2-byte CRC to the end of the packet after the payload. 0 = DISABLE : Disable hardware generation of CS (Host must calculate CS). 1 = ENABLE : Enable hardware generation of CS.
0	0x1	ECC_ENABLE: Enable hardware Error Correction Code (ECC) for Host packets Note: when ECC is disabled, Host is responsible for generating proper ECC byte for header 0 = DISABLE : Disable hardware generation of ECC (Host must calculate ECC). 1 = ENABLE : Enable hardware generation of ECC.

22.6.12 DSI_DSI_CONTROL_0

General DSI Control Register

Offset: 0x10 │ Byte Offset: 0x40 │ Read/Write: R/W │ Reset: 0x00000000 (0b00xxxxxxxxx0xx00xx00xx00xx000000)

Bit	Reset	Description
31	0x0	DSI_DBG_ENABLE: Control signal to turn off clock monitoring when enabled for debug, on every DSI byte clock debug signal toggle. Also TX CRC computation aid will turn on.
30	0x0	DFMT_16BPP_SWAP_EN: DSI specification supports different bit ordering (only 16 BPP) in command mode. Command Mode (Default) : 16BPP[15:0] -> B[4-0]G[5-0]R[4-0]; Command Mode (SWAP_EN) : 16BPP[15:0] -> G[2-0]B[4-0]R[4-0]G[5-3]
20	0x0	DSI_HS_CLK_CTRL: Control for the HS clock lane 0 = CONTINUOUS : HS clock is on all the time. 1 = TX_ONLY : HS clock is only active during HS transmissions.
17:16	0x0	DSI_VIRTUAL_CHANNEL: Virtual channel ID. The virtual channel is sent as part of the packet header and used to distinguish multiple displays.

Bit	Reset	Description
13:12	0x0	DSI_DATA_FORMAT: Pixel Data format transmitted. Note that although the pixel format is specified in the packet header ID for RGB data packets, this information is ignored by the hardware. Only the information used in this register is used in the construction of RGB data packets. 0 = BIT16P : 16 bpp RGB Packed. 2 bytes used per pixel 1 = BIT18NP : 18 bpp RGB Not-packed. 3 bytes used per pixel 2 = BIT18P : 18 bpp RGB Packed. 2.25 bytes used per pixel 3 = BIT24P : 24 bpp RGB Packed. 3 bytes used per pixel
9:8	0x0	VID_TX_TRIG_SRC: Controls the source of the trigger to start sending packets. 0 = SOL : Start of Line signal from the Display Controller. 1 = FIFO_LEVEL : How full the FIFO is. Level determined elsewhere. 2 = IMMEDIATE : Determined by a write to the DSI_VID_TRIGGER field of the DSI_TRIGGER register
5:4	0x0	DSI_NUM_DATA_LANES: Number of D-PHY data lanes used by Display for HS transmission. 0 = ONE : 1 data lane. 1 = TWO : 2 data lanes. 2 = THREE : 3 data lanes. 3 = FOUR : 4 data lanes.
3	0x0	VID_DCS_ENABLE: Enable for insertion of DCS commands during Display Controller generated packets. When enabled, the DCS commands defined in the LT3_DCS_CMD and LT5_DCS_CMD fields of the DSI_DCS_CMDS register will be inserted in long packets defined in packet sequence 3 and 5. 0 = DISABLE : No DCS commands will be inserted. 1 = ENABLE : DCS command IDs will be inserted as described above.
2	0x0	DSI_VID_SOURCE: Source of video pixels 0 = DISPLAY_0 : Pixels come from "display" 1 = DISPLAY_1 : Pixels come from "displayb"
1	0x0	DSI_VID_ENABLE: Video DSI Interface Enable 0 = DISABLE : Disable 1 = ENABLE : Enable
0	0x0	DSI_HOST_ENABLE: Host DSI Interface Enable 0 = DISABLE : Disable 1 = ENABLE : Enable

22.6.13 DSI_DSI_SOL_DELAY_0

Number of Byte-Clock Counts to Wait after Reception

Offset: 0x11 | Byte Offset: 0x44 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SOL_DELAY: Start Of Line before generating output packets.

22.6.14 DSI_DSI_MAX_THRESHOLD_0

Maximum Threshold Registers for DSI Related Packets

Offset: 0x12 | Byte Offset: 0x48 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	MAX_THRESHOLD: Start draining FIFO once this threshold is met. This register can be used for DBI mode when line packet data exceeds the size of the data FIFO.

22.6.15 DSI_DSI_TRIGGER_0

Manual Transmission Trigger Register

Offset: 0x13 | Byte Offset: 0x4c | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	DSI_HOST_TRIGGER: A 1 written to this bit will start host transmission when HOST_DSI_CONTROL.HOST_TX_TRIG_SRC is set to IMMEDIATE. Hardware auto clears on operation completion
0	X	DSI_VID_TRIGGER: A 1 written to this bit will start video transmission when DSI_CONTROL.VID_TX_TRIG_SRC is set to IMMEDIATE. Hardware auto clears on operation completion

22.6.16 DSI_DSI_TX_CRC_0

Transmission CRC

Offset: 0x14 | Byte Offset: 0x50 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TX_CRC: Long Packet CRC appended to the end of long packets. This CRC is that result of generating a CRC from all transmitted bytes. If DSI_HOST_ENABLE == ENABLE, then a CRC is generated from all bytes transmitted from the Host interface in Command Mode. If DSI_VID_ENABLE == ENABLE, then a CRC is generated from all bytes transmitted during a frame of video when in Video Mode. Note that the hardware will capture the CRC into a separate internal register so that it can continue to calculate the CRC for the next frame without having to wait for software to read the current result.

22.6.17 DSI_DSI_STATUS_0

DSI Status Register

Offset: 0x15 | Byte Offset: 0x54 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10	X	DSI_IDLE: Indicated that the DSI is IDLE.
9	X	LB_UNDERFLOW: Indicates that a Line buffer underflow event happened
8	X	LB_OVERFLOW: Indicates that a Line buffer overflow event happened
4:0	X	RD_FIFO_COUNT: Count of how many data words are left in the Host Read Data Return FIFO. Typically, software knows how much data to read from the DSI_RD_DATA register after a Read packet / BTA has been sent / requested, since these transactions are initiated by software. However, under error conditions, insufficient data may have been read from the FIFO. Software should therefore check this field to make sure it is 0 after reading all the information it expected to get.

22.7 Initialization Sequence Registers

22.7.1 DSI_DSI_INIT_SEQ_CONTROL_0

DSI Initialization Sequence Control

Offset: 0x1a | Byte Offset: 0x68 | Read/Write: R/W | Reset: 0x0000XX00 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
14:8	X	DSI_FRAME_INIT_BYTE_COUNT: Frame Initialization Sequence Byte Count. This parameter specifies the number of frame initialization sequence cycles to send. If programmed to 0, there is no frame initialization cycle generated. Valid programmable values: 0 to 64. Invalid programmable values: 65 to 127.
0	0x0	DSI_SEND_INIT_SEQUENCE: Send Initialization Sequence (IS) 0 = DISABLE 1 = ENABLE

22.7.2 DSI_DSI_INIT_SEQ_DATA_0_0

DSI Init Sequence Write Data 0

Offset: 0x1b | Byte Offset: 0x6c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_0: DSI Init Sequence Write Data bits 31:0

22.7.3 DSI_DSI_INIT_SEQ_DATA_1_0

DSI Init Sequence Write Data 1

Offset: 0x1c | Byte Offset: 0x70 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_1: DSI Init Sequence Write Data bits 31:0

22.7.4 DSI_DSI_INIT_SEQ_DATA_2_0

DSI Init Sequence Write Data 2

Offset: 0x1d | Byte Offset: 0x74 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_2: DSI Init Sequence Write Data bits 31:0

22.7.5 DSI_DSI_INIT_SEQ_DATA_3_0

DSI Init Sequence Write Data 3

Offset: 0x1e | Byte Offset: 0x78 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_3: DSI Init Sequence Write Data bits 31:0

22.7.6 DSI_DSI_INIT_SEQ_DATA_4_0

DSI Init Sequence Write Data 4

Offset: 0x1f | Byte Offset: 0x7c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_4: DSI Init Sequence Write Data bits 31:0

22.7.7 DSI_DSI_INIT_SEQ_DATA_5_0

DSI Init Sequence Write Data 5

Offset: 0x20 | Byte Offset: 0x80 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_5: DSI Init Sequence Write Data bits 31:0

22.7.8 DSI_DSI_INIT_SEQ_DATA_6_0

DSI Init Sequence Write Data 6

Offset: 0x21 | Byte Offset: 0x84 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_6: DSI Init Sequence Write Data bits 31-0

22.7.9 DSI_DSI_INIT_SEQ_DATA_7_0

DSI Init Sequence Write Data 7

Offset: 0x22 | Byte Offset: 0x88 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_7: DSI Init Sequence Write Data bits 31-0

22.8 Packet Sequence Registers

These registers allow the construction of arbitrary packet sequences associated with various video line types as sent from the Display Controller to the DSI block.

The first digit of the pair of digits in each field below is the sequence number, and the second digit denotes the packet number within each sequence. For example, field PKT_23_ID, defines the ID for packet 3 in sequence 2.

22.8.1 DSI_DSI_PKT_SEQ_0_LO_0

DSI Packet Sequence 0 LO Half

Offset: 0x23 | Byte Offset: 0x8c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
30	X	SEQ_0_FORCE_LP: For packet sequence 0, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_02_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_02_ID: Packet 2 Packet ID
22:20	X	PKT_02_SIZE: Packet 2 size pointer
19	X	PKT_01_EN: Packet 1 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_01_ID: Packet 1 Packet ID
12:10	X	PKT_01_SIZE: Packet 1 size pointer
9	X	PKT_00_EN: Packet 0 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_00_ID: Packet 0 Packet ID
2:0	X	PKT_00_SIZE: Packet 0 size pointer

22.8.2 DSI_DSI_PKT_SEQ_0_HI_0

DSI Packet Sequence 0 HI Half

Line Type 0 is associated with the first line in the frame and should contain a packet sequence that starts with a VS packet.

Offset: 0x24 | Byte Offset: 0x90 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
29	X	PKT_05_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_05_ID: Packet 5 Packet ID
22:20	X	PKT_05_SIZE: Packet 5 size pointer
19	X	PKT_04_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_04_ID: Packet 4 Packet ID
12:10	X	PKT_04_SIZE: Packet 4 size pointer
9	X	PKT_03_EN: Packet 3 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_03_ID: Packet 3 Packet ID
2:0	X	PKT_03_SIZE: Packet 3 size pointer

22.8.3 DSI_DSI_PKT_SEQ_1_LO_0

DSI Packet Sequence 1 LO Half

Line Type 1 is associated with the last line of Vertical Sync and should contain a packet sequence that starts with a VE packet.

Offset: 0x25 | Byte Offset: 0x94 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
30	X	SEQ_1_FORCE_LP: For packet sequence 1, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_12_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_12_ID: Packet 2 Packet ID
22:20	X	PKT_12_SIZE: Packet 2 size pointer
19	X	PKT_11_EN: Packet 1 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_11_ID: Packet 1 Packet ID
12:10	X	PKT_11_SIZE: Packet 1 size pointer
9	X	PKT_10_EN: Packet 0 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_10_ID: Packet 0 Packet ID
2:0	X	PKT_10_SIZE: Packet 0 size pointer

22.8.4 DSI_DSI_PKT_SEQ_1_HI_0

DSI Packet Sequence 1 HI Half

Offset: 0x26 | Byte Offset: 0x98 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
29	X	PKT_15_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_15_ID: Packet 5 Packet ID
22:20	X	PKT_15_SIZE: Packet 5 size pointer
19	X	PKT_14_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_14_ID: Packet 4 Packet ID
12:10	X	PKT_14_SIZE: Packet 4 size pointer
9	X	PKT_13_EN: Packet 3 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_13_ID: Packet 3 Packet ID
2:0	X	PKT_13_SIZE: Packet 3 size pointer

22.8.5 DSI_DSI_PKT_SEQ_2_LO_0

DSI Packet Sequence 2 LO Half

Line Type 2 is associated with any vertical blank line except the first one after active and should contain a packet sequence that starts with an HS packet.

Offset: 0x27 │ Byte Offset: 0x9c │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
30	X	SEQ_2_FORCE_LP: For packet sequence 2, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_22_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_22_ID: Packet 2 Packet ID
22:20	X	PKT_22_SIZE: Packet 2 size pointer
19	X	PKT_21_EN: Packet 1 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_21_ID: Packet 1 Packet ID
12:10	X	PKT_21_SIZE: Packet 1 size pointer
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

22.8.6 DSI_DSI_PKT_SEQ_2_HI_0

DSI Packet Sequence 2 HI Half

Offset: 0x28 │ Byte Offset: 0xa0 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
29	X	PKT_25_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_25_ID: Packet 5 Packet ID
22:20	X	PKT_25_SIZE: Packet 5 size pointer
19	X	PKT_24_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_24_ID: Packet 4 Packet ID
12:10	X	PKT_24_SIZE: Packet 4 size pointer
9	X	PKT_23_EN: Packet 3 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_23_ID: Packet 3 Packet ID
2:0	X	PKT_23_SIZE: Packet 3 size pointer

22.8.7 DSI_DSI_PKT_SEQ_3_LO_0

DSI Packet Sequence 3 LO Half

Line Type 3 is associated with any active line except the first one and should contain a packet sequence that starts with an HS packet and includes an RGB data packet.

Offset: 0x29 │ Byte Offset: 0xa4 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
30	X	SEQ_3_FORCE_LP: For packet sequence 3, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_32_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_32_ID: Packet 2 Packet ID
22:20	X	PKT_32_SIZE: Packet 2 size pointer
19	X	PKT_31_EN: Packet 1 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_31_ID: Packet 1 Packet ID
12:10	X	PKT_31_SIZE: Packet 1 size pointer
9	X	PKT_30_EN: Packet 0 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_30_ID: Packet 0 Packet ID
2:0	X	PKT_30_SIZE: Packet 0 size pointer

22.8.8 DSI_DSI_PKT_SEQ_3_HI_0

DSI Packet Sequence 3 HI Half

Offset: 0x2a │ Byte Offset: 0xa8 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

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

22.8.9 DSI_DSI_PKT_SEQ_4_LO_0

DSI Packet Sequence 4 LO Half

Line Type 4 is associated with the first vertical blanking line after the last active line and should contain a packet sequence that starts with an HS packet. Ordinarily, this packet sequence should be identical to sequence 2.

Offset: 0x2b │ Byte Offset: 0xac │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

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

22.8.10 DSI_DSI_PKT_SEQ_4_HI_0

DSI Packet Sequence 4 HI Half

Offset: 0x2c │ Byte Offset: 0xb0 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
29	X	PKT_45_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_45_ID: Packet 5 Packet ID
22:20	X	PKT_45_SIZE: Packet 5 size pointer
19	X	PKT_44_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_44_ID: Packet 4 Packet ID
12:10	X	PKT_44_SIZE: Packet 4 size pointer
9	X	PKT_43_EN: Packet 3 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_43_ID: Packet 3 Packet ID
2:0	X	PKT_43_SIZE: Packet 3 size pointer

22.8.11 DSI_DSI_PKT_SEQ_5_LO_0

DSI Packet Sequence 5 LO Half

Line Type 5 is associated with the first active line. It should contain a packet sequence that starts with an HS packet and includes an RGB data packet. Ordinarily, this packet sequence should be identical to sequence 3.

Offset: 0x2d │ Byte Offset: 0xb4 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
30	X	SEQ_5_FORCE_LP: For packet sequence 5, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_52_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
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

22.8.12 DSI_DSI_PKT_SEQ_5_HI_0

DSI Packet Sequence 5 HI Half

Offset: 0x2e │ Byte Offset: 0xb8 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
29	X	PKT_55_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_55_ID: Packet 5 Packet ID
22:20	X	PKT_55_SIZE: Packet 5 size pointer
19	X	PKT_54_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_54_ID: Packet 4 Packet ID
12:10	X	PKT_54_SIZE: Packet 4 size pointer
9	X	PKT_53_EN: Packet 3 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_53_ID: Packet 3 Packet ID
2:0	X	PKT_53_SIZE: Packet 3 size pointer

22.9 DCS Command and Packet Length Registers

22.9.1 DSI_DSI_DCS_CMDS_0

DCS command IDs used for Line Types 3 and 5. These command IDs are inserted at the start of the data payload of DCS long packets when the Display Controller is being used to transmit DCS commands.

Offset: 0x33 | Byte Offset: 0xcc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
15:8	X	LT5_DCS_CMD: DCS command for Line Type 5.
7:0	X	LT3_DCS_CMD: DCS command for Line Type 3.

22.9.2 DSI_DSI_PKT_LEN_0_1_0

DSI Packet Lengths 0 and 1

Offset: 0x34 | Byte Offset: 0xd0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LENGTH_1: Packet length 1 (in bytes)
15:0	X	LENGTH_0: Packet length 0 (in bytes)

22.9.3 DSI_DSI_PKT_LEN_2_3_0

DSI Packet Lengths 2 and 3

Offset: 0x35 | Byte Offset: 0xd4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LENGTH_3: Packet length 3 (in bytes)
15:0	X	LENGTH_2: Packet length 2 (in bytes)

22.9.4 DSI_DSI_PKT_LEN_4_5_0

DSI Packet Lengths 4 and 5

Offset: 0x36 | Byte Offset: 0xd8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LENGTH_5: Packet length 5 (in bytes)
15:0	X	LENGTH_4: Packet length 4 (in bytes)

22.9.5 DSI_DSI_PKT_LEN_6_7_0

DSI Packet Lengths 6 and 7

Offset: 0x37 | Byte Offset: 0xdc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LENGTH_7: Packet length 7 (in bytes)
15:0	X	LENGTH_6: Packet length 6 (in bytes)

22.10 Physical Interface Timing Registers

The DSI PHY timing registers must be initialized before the interface is enabled. Based on the equations provided in the table below, all PHY timing parameters should be programmed in the timing registers. T_{byte} refers to just the DSI controller clock.

Table 102 D-PHY Timing Values

Parameter	MIPI D-PHY Spec Value	Programmed Value
THS-PREPARE	(40 ns + 4*UI , 85 ns + 6*UI)	Min - (40 + (4 T_{bit}))) / T_{byte} Max - ((85 + 6 T_{bit})) / T_{byte}
THS-ZERO	105 ns + 6*UI min.	(105 + (6 T_{bit})) / T_{byte} -2
THS-TRAIL	max(8*UI, 60 ns + 4*UI)	3 + max(8 T_{bit}, 60 + 4 T_{bit}) / T_{byte}
THS-EXIT	100 ns min.	100 / T_{byte}
TLPX	50 ns min.	50 / T_{byte}
TCLK-ZERO	300 ns along with TCLK-PREPARE	260/ T_{byte}
TCLK-PREPARE	(38 ns, 95 ns)	THS-PREPARE timing should guarantee to meet both these timing requirements
TCLK-POST	60 ns + 52*UI min.	(60 + (52 T_{bit})) / T_{byte} - 2
TCLK-TRAIL	60 ns min.	60 / T_{byte}
TCLK-PRE	8 UI	8 T_{bit}
TTA-GO	4 * TLPX	3 * TLPX
TTA-SURE	(TLPX, 2 * TLPX)	TLPX - 1
TTA-GET	5 * TLPX	4 * TLPX
TWAKEUP	1 ms	1 ms/10 ns x 512 {in terms of 512 byte clocks}

Notes:

- T_{byte} = Period of one byte clock in nanoseconds.
- T_{bit} = Period of one bit time in nanoseconds. (T_{bit} = T_{byte} / 8)
- All figures and time periods are in nanoseconds.
- All timing register values are multiples of byte clock period, T_{byte}.
- All fractional results are truncated. No rounding.

22.10.1 DSI_DSI_PHY_TIMING_0_0

DSI D-PHY Timing Register 0

Offset: 0x3c │ Byte Offset: 0xf0 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	DSI_THSDEXIT: Time to drive LP11 after HS **Note:** Valid programmable values for all the fields in this register (i.e., DSI_THSPREPR/DSI_TDATZERO/DSI_THSTRAIL/DSI_THSDEXIT) are 0 to 254. Invalid value is 255.
23:16	X	DSI_THSTRAIL: Time to drive HS flipped bit at EOT
15:8	X	DSI_TDATZERO: Time to drive HS0 before SOT

Bit	Reset	Description
7:0	X	DSI_THSPREPR: Time to drive LP00 before HS data

22.10.2 DSI_DSI_PHY_TIMING_1_0

DSI D-PHY Timing Register 1

Offset: 0x3d | Byte Offset: 0xf4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	DSI_TCLKTRAIL: Time to drive HS0 before the clock goes to LP1 **Note:** Valid programmable values for all the fields in this register (i.e., DSI_TTLPX/DSI_TCLKZERO/DSI_TCLKPOST/ DSI_TCLKTRAIL) are 0 to 254 Invalid value is 255.
23:16	X	DSI_TCLKPOST: Time to drive clock after the last HS data
15:8	X	DSI_TCLKZERO: Time to drive LP00 before HS clock
7:0	X	DSI_TTLPX: LP period

22.10.3 DSI_DSI_PHY_TIMING_2_0

DSI D-PHY Timing Register 2

Offset: 0x3e | Byte Offset: 0xf8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
23:16	X	DSI_TCLKPREPARE: Time to drive LP0 before CLK_ZERO starts off on the clock lane. **Note:** Valid programmable values for the following register fields (i.e., DSI_TCLKPRE/DSI_TCLKPREPARE) are 0 to 254. Invalid value is 255
15:8	X	DSI_TCLKPRE: Time to run clock before enabling data lane
7:0	X	DSI_TWAKEUP: LP period when exiting ULPM, in units of 512 byte clocks.

22.10.4 DSI_DSI_BTA_TIMING_0

DSI D-PHY Bus-Turn-Around Timing

Offset: 0x3f | Byte Offset: 0xfc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	DSI_TPKTBTA: Programmable Time delay between end of Host packet transmission and generation of Pkt BTA in PKT_BTA mode (i.e., HOST_DSI_CONTROL[2]) - Generate BTA at the end of Host packets **Note**: Valid programmable values for all the fields in this register (i.e., DSI_TTAGO/DSI_TTASURE/DSI_TTAGET/ DSI_TPKTBTA) are 0 to 254. Invalid value is 255.
23:16	X	DSI_TTAGET: Time to Drive LP00 at end of BTA (5 * TTLPX)
15:8	X	DSI_TTASURE: Time to Receive LP00 at end of BTA (2 * TTLPX)
7:0	X	DSI_TTAGO: Time to drive LP00 at start of BTA (4 * TTLPX)

22.11 Contention Recovery Timers

These registers control the length of time - in units of 512 DSI byte clocks – that can elapse before the hardware will decide that a bus contention error has occurred.

There are several counters to deal with various forms of error that may occur. For details, refer to the MIPI DSI Specification, section 7.2.2.

22.11.1 DSI_DSI_TIMEOUT_0_0

DSI Time Out Terminal Count Register 0

Offset: 0x44 │ Byte Offset: 0x110 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LRXH_TO: Low Power Receive (Host) Time Out terminal count
15:0	X	HTX_TO: High Speed Transmit Time Out terminal count

22.11.2 DSI_DSI_TIMEOUT_1_0

DSI Time Out Terminal Count Register 1

Offset: 0x45 │ Byte Offset: 0x114 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	PR_TO: Peripheral Reset duration.
15:0	X	TA_TO: Turn Around Time Out terminal count

22.11.3 DSI_DSI_TO_TALLY_0

DSI Time Out Tally Register

Each time one of the time-out counters reaches its terminal count, it will increment the associated tally register.

Offset: 0x46 │ Byte Offset: 0x118 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	R/W	Reset	Description
24	RO	X	P_RESET_STATUS: Peripheral Reset time out status 0 = IN_RESET 1 = READY
23:16	RW	X	TA_TALLY: Turn Around time out tally
15:8	RW	X	LRXH_TALLY: LP Rx time out tally
7:0	RW	X	HTX_TALLY: HS Tx time out tally

22.12 Physical Pad Control Registers

22.12.1 DSI_PAD_CONTROL_0

DSI PHY Configuration Register

Software can program this register to work with different panels.

Offset: 0x4b │ Byte Offset: 0x12c │ Read/Write: R/W │ Reset: 0x010f010f (0bxxxxxxx1xxxx1111xxxxxxx1xxxx1111)

Bit	Reset	Description
24	0x1	DSI_PAD_PULLDN_CLK_ENAB:1= Enable pad pulldown for clock bit at power on
19:16	0xf	DSI_PAD_PULLDN_ENAB:1= Enable pad pulldown for data bits at power on
8	0x1	DSI_PAD_PDIO_CLK: Power down for clock bit, drivers, receivers, and contention detectors
3:0	0xf	DSI_PAD_PDIO: Power down for data bit, drivers, receivers, and contention detectors

22.12.2 DSI_PAD_CONTROL_CD_0

Contention Detection Logic Enable Signals

Offset: 0x4c │ Byte Offset: 0x130 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxx000xxxxxxx0xxxx0000)

Bit	Reset	Description
18:16	0x0	DSI_PAD_CDDNADJ: Level adjust on low limit of detection. Tie to 0 (DSI emu, miniTMC control). 000 = 0.3V 001 = 0.375V 010 = 0.45V 011 = 0.525V 100 = 0.3V 101 = 0.225V 110 = 0.15V 111 = 0.075V
8	0x0	DSI_PAD_CD_EN_CLK: Clock bit contention detector enable. 1 = Enable
3:0	0x0	DSI_PAD_CD_EN: Data bits contention detector enable. 1 = EnableCD_EN_[1] = 0 in DSI.

22.12.3 DSI_PAD_CD_STATUS_0

Contention Detection Status from MIPI PAD

Offset: 0x4d │ Byte Offset: 0x134 │ Read/Write: RO │ Reset: 0x000X0X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18	X	DSI_PAD_CDN_CLK
16	X	DSI_PAD_CDP_CLK
11:8	X	DSI_PAD_CDN
3:0	X	DSI_PAD_CDP

22.12.4 DSI_DSI_VID_MODE_CONTROL_0

Host Command Packet During Video Mode

Offset: 0x4e │ Byte Offset: 0x138 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:1	0x0	DSI_LINE_TYPE: Vertical blank. LINE TYPE on which the host command packet is to be transmitted, i.e., valid values are 0/1/2/4 0 = ZERO : Line type 0 1 = ONE : Line type 1 2 = TWO : Line type 2 3 = THREE : NA: Host command packets on line type 3 are invalid 4 = FOUR : Line type 4 5 = FIVE : NA: Host command packets on line type 5 are invalid 6 = SIX : NA: Host command packets on line type 6 are invalid 7 = SEVEN : NA: Host command packets on line type 7 are invalid
0	0x0	DSI_CMD_PKT_VID_ENABLE: 0 = DISABLE : Disable host command packet during video mode 1 = ENABLE : Enable host command packet during video mode

22.12.5 DSI_PAD_CONTROL_1_0

DSI PHY Configuration Register

Software can program this register to work with different panels.

Offset: 0x4f │ Byte Offset: 0x13c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx000x000x000x000)

Bit	Reset	Description
14:12	0x0	DSI_PAD_OUTADJ3: Input delay trimmer for data bit 3. Each tap delays 40 ps
10:8	0x0	DSI_PAD_OUTADJ2: Input delay trimmer for data bit 2. Each tap delays 40 ps
6:4	0x0	DSI_PAD_OUTADJ1: Input delay trimmer for data bit 1. Each tap delays 40 ps
2:0	0x0	DSI_PAD_OUTADJ0: Input delay trimmer for data bit 0.Each tap delays 40 ps

22.12.6 DSI_PAD_CONTROL_2_0

DSI PHY Configuration Register

Software can program this register to work with different panels.

Offset: 0x50 │ Byte Offset: 0x140 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxx000x000x000x000x000)

Bit	Reset	Description
18:16	0x0	DSI_PAD_SLEWUPADJ: Pull-up slew rate adjust. 000 = 011: slew rate increases 100 = 000, 100 =111: skew rate decreases
14:12	0x0	DSI_PAD_SLEWDNADJ: Pull-down slew rate adjust. 000 = 011, slew rate increases 100 = 000, 100 = 111: skew rate decreases.
10:8	0x0	DSI_PAD_LPUPADJ: Driver pull-up impedance control. 00 = 130 ohms, default, 01 = 110 ohms 10 = 130 ohms 11 = 150 ohms

Bit	Reset	Description
6:4	0x0	DSI_PAD_LPDNADJ: Driver pull-down impedance control. 00 = 130 ohms, default 01 = 110 ohms 10 = 130 ohms, same as 0 011 = 150 ohms
2:0	0x0	DSI_PAD_OUTADJCLK: Output trimmer delay for clock bit. Each tap delays 40 ps

22.12.7 DSI_PAD_CONTROL_3_0

DSI PHY Configuration Register

Software can program this register to work with different panels.

Offset: 0x51 │ Byte Offset: 0x144 │ Read/Write: R/W │ Reset: 0x10000000 (0bxxx1xxxxxxxxxxx0xx00xx00xx00xx00)

Bit	Reset	Description
28	0x1	DSI_PAD_PDVCLAMP: Power down regulator which supplies current to serializer/deserializer logic. Active high, 1=power down
16	0x0	DSI_PAD_BANDWD_IN: Increase bandwidth of differential receiver
13:12	0x0	DSI_PAD_PREEMP_PD_CLK: Clock bit HS driver pull down pre-emphasis. 00 = no_preemphasis 11= max
9:8	0x0	DSI_PAD_PREEMP_PU_CLK: Clock bit HS driver pull up pre-emphasis. 00 = no_preemphasis 11= max
5:4	0x0	DSI_PAD_PREEMP_PD: Data bit HS driver pull down pre-emphasis. 00 = no_preemphasis 11= max
1:0	0x0	DSI_PAD_PREEMP_PU: Data bit HS driver pull up pre-emphasis. 00 = no_preemphasis 11= max

22.12.8 DSI_PAD_CONTROL_4_0

DSI PHY Configuration Register

Software can program this register to work with different panels.

Offset: 0x52 │ Byte Offset: 0x148 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxx0xxxx0000xxx0xxxx0000xxx0xxx0)

Bit	Reset	Description
28	0x0	DSI_PAD_HS_BSO_CLK:1= Enables BIAS and power regulators on for HS mode
23:20	0x0	DSI_PAD_HS_BSO:1= Enables BIAS and power regulators on for HS mode
16	0x0	DSI_PAD_LP_BSO_CLK:1= Enables BIAS and power regulators on for LP mode
11:8	0x0	DSI_PAD_LP_BSO:1= Enables BIAS and power regulators on for LP mode
4	0x0	DSI_PAD_TXBW_EN:1= Increase bandwidth of output driver, default=0
0	0x0	DSI_PAD_REV_CLK:1= Reverse clock polarity

22.12.9 DSI_DSI_GANGED_MODE_CONTROL_0

Offset: 0x53 | Byte Offset: 0x14c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	DSI_GANGED_MODE_EN: 0 = DISABLE : Ganged mode transaction disabled 1 = ENABLE : Ganged mode transaction enabled

22.12.10 DSI_DSI_GANGED_MODE_START_0

Offset: 0x54 | Byte Offset: 0x150 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxx0000000000000)

Bit	Reset	Description
12:0	0x0	DSI_GANGED_START_POINTER: Start pointer for indicating the start of partial active valid pixel data to be latched from the valid pixels of the display controller in Left-Right/Even-Odd ganged mode. Note: Program to the default value/zero when DSI_GANGED_MODE_EN = DISABLE (non ganged mode).

22.12.11 DSI_DSI_GANGED_MODE_SIZE_0

Offset: 0x55 | Byte Offset: 0x154 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0000000000000xxx0000000000000)

Bit	Reset	Description
28:16	0x0	DSI_GANGED_VALID_LOW_WIDTH: Width of Partial Inactive/Ignored pixel data from the valid pixels of the display controller in Left-Right/Even-Odd ganged mode. Note: Program to the default value/zero when DSI_GANGED_MODE_EN = DISABLE (non ganged mode).
12:0	0x0	DSI_GANGED_VALID_HIGH_WIDTH: Width of Partial Active valid pixel data latched from the valid pixels of display controller in Left-Right/Even-Odd ganged mode. Note: Program to the default value/zero when DSI_GANGED_MODE_EN = DISABLE (non ganged mode).

22.12.12 DSI_DSI_RAW_DATA_BYTE_COUNT_0

Offset: 0x56 | Byte Offset: 0x158 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	DSI_RAW_DATA_BYTE_COUNT: Host RAW DATA byte count specifies the total number of bytes to send when "HOST_DSI_CONTROL[6] -> RAW_DATA" enabled

22.12.13 DSI_DSI_ULTRA_LOW_POWER_CONTROL_0

Ultra Low Power Sequence Control Register

Offset: 0x57 | Byte Offset: 0x15c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:8	0x0	DSI_ULTRA_LOW_POWER_DATA_LANE3: Ultra Low Power 0 = NORMAL 1 = ENTER_ULPM 2 = EXIT_ULPM
7:6	0x0	DSI_ULTRA_LOW_POWER_DATA_LANE2: Ultra Low Power 0 = NORMAL 1 = ENTER_ULPM 2 = EXIT_ULPM

Bit	Reset	Description
5:4	0x0	DSI_ULTRA_LOW_POWER_DATA_LANE1: Ultra Low Power 0 = NORMAL 1 = ENTER_ULPM 2 = EXIT_ULPM
3:2	0x0	DSI_ULTRA_LOW_POWER_DATA_LANE0: Ultra Low Power 0 = NORMAL 1 = ENTER_ULPM 2 = EXIT_ULPM
1:0	0x0	DSI_ULTRA_LOW_POWER_CLK_LANE: Ultra Low Power 0 = NORMAL 1 = ENTER_ULPM 2 = EXIT_ULPM

22.12.14 DSI_DSI_INIT_SEQ_DATA_8_0

DSI Init Sequence Write Data 8

Offset: 0x58 | Byte Offset: 0x160 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_8: DSI Init Sequence Write Data bits 31:0

22.12.15 DSI_DSI_INIT_SEQ_DATA_9_0

DSI Init Sequence Write Data 9

Offset: 0x59 | Byte Offset: 0x164 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_9: DSI Init Sequence Write Data bits 31:0

22.12.16 DSI_DSI_INIT_SEQ_DATA_10_0

DSI Init Sequence Write Data 10

Offset: 0x5a | Byte Offset: 0x168 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_10: DSI Init Sequence Write Data bits 31:0

22.12.17 DSI_DSI_INIT_SEQ_DATA_11_0

DSI Init Sequence Write Data 11

Offset: 0x5b | Byte Offset: 0x16c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_11: DSI Init Sequence Write Data bits 31:0

22.12.18 DSI_DSI_INIT_SEQ_DATA_12_0

DSI Init Sequence Write Data 12

Offset: 0x5c | Byte Offset: 0x170 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_12: DSI Init Sequence Write Data bits 31:0

22.12.19 DSI_DSI_INIT_SEQ_DATA_13_0

DSI Init Sequence Write Data 13

Offset: 0x5d | Byte Offset: 0x174 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_13: DSI Init Sequence Write Data bits 31:0

22.12.20 DSI_DSI_INIT_SEQ_DATA_14_0

DSI Init Sequence Write Data 14

Offset: 0x5e | Byte Offset: 0x178 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_14: DSI Init Sequence Write Data bits 31:0

22.12.21 DSI_DSI_INIT_SEQ_DATA_15_0

DSI Init Sequence Write Data 15

Offset: 0x5f | Byte Offset: 0x17c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_15: DSI Init Sequence Write Data bits 31:0

23.0 HIGH-DEFINITION MULTIMEDIA INTERFACE

The High-Definition Multimedia Interface (HDMI) receives pixels from the Display Controller and audio from the HDA. It combines and transmits them together.

23.1 Features

- HARDWARE_N feature enabled
- TMDS I/O macro with controls
- Interlaced video format support
- YUV444 output (BT.601 and BT.709 formats)
- HDMI in power-gated partition SOR
- Display CMU, allowing color management
- Output range scaling for RGB
- Vendor-specific infoframe supported
- Audio features for HD audio include:
    - 2, 6, or 8 channels of LPCM (i.e., uncompressed linear PCM)
    - 16, 20, or 24 bits per sample
    - 32, 44.1, 48, 88.2, 96, 176.4, 192 K samples/second

23.2 Programming Guidelines

The overall sequence to start up HDMI is:

- Program power, pin mux, clocks, resets, etc.
- Copy HDCP keys from KFUSE to HDMI
- Program display timing registers, etc.
- Program HDMI registers and Serial Output Resource (SOR) sequencer
- Start HDMI SOR
- Start display

Since audio is entirely asynchronous to HDMI or the display, the HDA module can be started any time.

23.2.1 Power

Turn on the necessary power supplies (these need to be enabled in the PMIC via I^2C):

- AVDD_HDMI_PLL
- AVDD_HDMI
- Onboard 5V for DDC/HDMI_INT/CEC

Set HDMI I/O logic to exit deep power down mode: make sure the HDMI bit in the APBDEV_PMC_IO_DPD_STATUS_0 register and the HV bit in the APBDEV_PMC_IO_DPD2_STATUS_0 register were cleared (see the registers in the PMC section).

23.2.2 DDC / I2C

DDC I^2C is used for E-EDID reading, HDCP, and other communication with downstream HDMI device. I^2C has no connection to HDMI block; the platform defines which I^2C engine is connected to the DDC.

23.2.3 Hot Plug

HDMI_INT is a pad that feeds the GPIO block and the PMC. It has no connection to the HDMI block. It operates as a general-purpose I/O pin.

23.2.4 Clocks

HDMI uses two main clocks:

- hdmi_clk: variable from 25.2 MHz to 300 MHz based on the video pixel rate. The frequency and source must be identical to the display's pixel clock. (i.e., display and HDMI clocks are isochronous).
- hda2hdmi_bitclk: fixed 24 MHz HD Audio bit clock. Same source as HDA Controller's bitclk. The HDA Controller clock enable should not affect it.

And three others:

- display2hdmi_pclk and displayb2hdmi_pclk: Copy of display pixel clocks for display2hdmi and displayb2hdmi pixel buses. Only used to write display data into FIFOs.
- host1x2hdmi_clk: Copy of host1x clock for host interface; only used to write host1x transactions into the FIFO.

All three main clocks have common first-level gate: the CAR CLK_ENB_HDMI register. hdmi_audio_clk has a second level gate controlled by hardware, which is gated.

The display controllers and HDMI have separate clock source registers. However, they must be programmed to the same source and the same frequency. Typical settings for a 13 MHz oscillator using PLLD are listed in the following table:

Pixel Clk (MHz)	PLLD Frequency	PLL M	PLL N	HDMI_ CLK_ DIVISOR	Display SHIFT_ CLK_ DIVIDER	HDMI_CLK_SRC	DISP1_CLK_SRC
27	216/2	13	216	0x6 (/4)	0x6 (/4)	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
297	594/2	13	594	0x0 (/1)	0x0 (/1)	PLLD2_OUT0	PLLD2_OUT0

In addition, the HDA module must use the same oscillator as display and HDMI, since audio and video clock frequencies must be a fixed, known ratio. For HDA Audio, HDA2CODEC_2X clock must be enabled and configured for 48 MHz.

23.2.5 Display Timing

Frame timing is controlled by the display. HDMI has certain requirements, however, imposed by both the module itself and the HDMI standards.

HDMI gets the following signals from the display:

- h_blank – derived in hardware from h_ref_to_sync, h_sync_width, h_back_porch, h_disp_active, h_front_porch
- v_blank – derived in hardware from v_ref_to_sync, v_sync_width, v_back_porch, v_disp_active, v_front_porch
- h_sync – derived from h_ref_to_sync, h_sync_width
- v_sync – derived from v_ref_to_sync, v_sync_width
- video_preamble – derived from h_pulse2_control, h_pulse2_position.

EIA (CEA-861-B) Formats	ref_to_sync	sync_width	back_porch	disp_active	front_porch
480P_60 (2,3)					
horizontal	1	62	60	720	16
vertical	1	6	30	480	9
720P_60 (4)					
horizontal	1	40	220	1280	110
vertical	1	5	20	720	5
720P_50 (19)					
horizontal	1	40	220	1280	440
vertical	1	5	20	720	5
640x480_60 (1)					
horizontal	1	96	48	640	16
vertical	1	2	33	480	10
576p_50 (17/18)					
horizontal	1	64	68	720	12
vertical	1	5	39	576	5
1080p_60 (16)					
horizontal	1	44	148	1920	88
vertical	1	5	36	1080	4
2160p_30					
horizontal	1	88	296	3840	176
vertical	1	10	72	2160	8
1080i_60 (5)					
horizontal	1	44	148	1920	88
vertical	1	5	15	540	2

The video_preamble uses h_pulse2, with the following settings:

disp_signal_options0.h_pulse2_enable	ENABLE
h_pulse2_control.h_pulse2_mode	NORMAL
h_pulse2_control.h_pulse2_polarity	HIGH
h_pulse2_control.h_pulse2_v_qual	VACTIVE
h_pulse2_control.h_pulse2_last_end	END_A
h_pulse2_position.h_pulse2_start_a	h_ref_to_sync + h_sync_width + h_back_porch - 10
h_pulse2_position.h_pulse2_end_a	h_pulse2_start_a + 8

Miscellaneous registers required:

DISP_TIMING_OPTIONS.VSYNC_H_POSITION	1
DISP_CLOCK_CONTROL.PIXEL_CLK_DIVIDER	PCD1
DISP_COLOR_CONTROL.DITHER_CONTROL	DISABLE
DISP_WIN_OPTIONS.HDMI_ENABLE	ENABLE

23.2.6 Display Datapath

23.2.6.1 Dither

Dithering should be disabled, since HDMI uses a full 8 bits per component.

23.2.6.2 Component Range

For most HDMI resolutions (i.e., everything except VGA 640x480), R/G/B should be scaled to lie in the [16, 235] nominal range; See the HDMI specification, section 6.6. The range scaling can be done in either of two places. For Tegra® 4 and Tegra 4i processor compatibility; CMU is recommended. For Tegra K1 code, CSC2 is recommended.

1. In the CMU block:

```
proc cmu_lut1_flt2fix {flt} {
    set fx [expr floor($flt * 4095.0 + 0.5)]
    if {$fx > 4095} {
        error "cmu_lut1_flt2fix out of range input $flt"
    }
    return $fx
}
proc cmu_lut2_flt2fix {flt} {
    set fx [expr floor($flt * 255.0 + 0.5)]
    if {$fx > 255} {
        error "cmu_lut1_flt2fix out of range input $flt"
    }
    return $fx
}
proc srgb_ungamma {x args} {
    if {$x <= 0.03928} {
        set y [expr $x/12.92]
    } else {
        set y [expr pow(($x + 0.055)/1.055, 2.4)]
    }
    return [cmu_lut1_flt2fix $y]
}
proc rec709_gamma_offset16 {x args} {
    if {$x <= 0.018} {
        set y [expr $x * 4.5]
    } else {
        set y [expr 1.099 * pow($x, 0.45) - 0.099]
    }
    set lut2 [expr 16 + [cmu_lut2_flt2fix $y]]
    return [expr min(255, max($lut2, 0))]
}

#
# program CMU to convert to limited-range RGB with Rec. 709 gamma.
# Display pipe should be using full-range RGB.
# [0..255] is mapped to [16,235].
# NOTE: sRGB primaries and whitebpoint are same as 709, so matrix is just scale.
# We could do everything in LUT as well.
#
proc cmu_to_rec709_rgb219 {} {
    set scale [expr 219.0 / 255.0]
    set mat "
```

```
        $scale 0.0    0.0    \
        0.0    $scale 0.0    \
        0.0    0.0    $scale \
    "
    cmu_lut1 srgb_ungamma
    cmu_csc $mat
    cmu_lut2 rec709_gamma_offset16
    DISP_COLOR_CONTROL.CMU_ENABLE ENABLE
}
```

2. In the CSC2 block:

```
CSC2_CONTROL.OUTPUT_COLOR_SELECT = RGB
CSC2_CONTROL.LIMIT_RGB_COLOR = ENABLE
```

For most HDMI modes, pixel values 0 and 255 must be removed; HDMI's NV_PDISP_INPUT_CONTROL.ARM_VIDEO_RANGE = LIMITED enables this.

23.2.6.3 Gamma

Most content uses sRGB gamma which is not quite the same as Rec. 709 gamma used by HDMI for most modes. Post-composite gamma can be changed in the CMU. The above code illustrates these.HDMI basics.

23.2.7 HDMI Basics

- Select display a/b source with NV_PDISP_HDMI_SRC_SELECT
- If appropriate to the platform, program HDCP timing dividers, based on the HDMI pixel clock. For Tegra K1 platforms, ROM I^2C clock frequency should be 200 KHz.
- Program SOR reference clock, based on the HDMI pixel clock – see the NV_PDISP_SOR_REFCLK register description in this section. For example,

```
dispclk_div_8_2 = int(dispclk_freq / 1000000.0 * 4)
NV_PDISP_SOR_REFCLK.DIV_INT = dispclk_div_8_2 >> 2
NV_PDISP_SOR_REFCLK.DIV_FRAC = dispclk_div_8_2 & 0x3
```

- Program SOR sequences for power up and power down.

23.2.8 TMDS Macro Configuration

The table below provides register values for various modes. Registers not listed should use their initial (reset) values.

HDMI Register	480p (27 MHz)	720p (74.25 MHz)	1080p (148.5 MHz)	1080i (74.25 MHz)	2160p (297 MHz)
HDMI_NV_PDISP_SOR_PLL0_0	0x01003010	0x01003110	0x01003310	0x01003110	0x01003F10
HDMI_NV_PDISP_SOR_PLL1_0	0x00301B00	0x00301500	0x00301500	0x00301500	0x00300F00
HDMI_NV_PDISP_SOR_LANE_DRIVE_CURRENT_0	0x1F1F1F1F	0x2C2C2C2C	0x33333333	0x2C2C2C2C	0x37373737
HDMI_NV_PDISP_PE_CURRENT_0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
HDMI_NV_PDISP_SOR_IO_PEAK_CURRENT_0	0x03030303	0x07070707	0x0C0C0C0C	0x07070707	0x17171717
HDMI_NV_PDISP_SOR_PAD_CTLS0_0	0x800034BB	0x800034BB	0x800034BB	0x800034BB	0x800036BB
CLK_RST_CONTROLLER_PLLD_MISC_0	0x40400820	0x40400820	0x40400820	0x40400820	0x40400F20

Note: NV_PDISP_SOR_LANE_DRIVE_CURRENT. LANE0..3 are only used if NV_PDISP_SOR_LANE_DRIVE_CURRENT.FUSE_OVERRIDE is TRUE. HDMI TMDS calibration can be performed and burned into fuses, but are not currently.

23.2.9 HDA Controller Initialization

Basic operations:

- HDA IPFS initialization, FPCI and HDA configuration, and memory spaces allocated
- Deassert soft reset, bring up link. AZA.GCTL.CRST = DEASSERT, and wait for reset completion by spinning until CRST = DEASSERT. Wait 4 frame times (poll AZA.WALCLK for 2000 ticks to elapse)
- Verify internal codec (and external if desired) have been detected by reading AZA.WAKEEN.SDIWAKE3 for internal codec, and SDIWAKE1 for external.

23.2.10 HDA Audio Codec Initialization

When HDA is the audio source, the codec must be initialized before audio can flow. The internal codec uses industry-standard verbs, so in theory, the out-of-box driver should correctly discover and configure the codec. See the HD Audio specification. Below is a summary of the necessary verbs:

1. AOC widget: SET_CONV_STREAM_CHAN with desired output stream ID
2. AOC widget: SET_CONV_FMT to desired audio format (sampling frequency, sample depth, number of channels)
3. AOC widget: SET_DIGITAL_CONV_CONTROL1
   - Bit 0: 1 (DigEn=1)
   - Bit 3..6: channel status pre/copy/audio/pro bits
   - Bit 7: channel status CC bit 7 (called GEN_LEVEL)
4. AOC widget: SET_DIGITAL_CONV_CONTROL2:
   - Bit 6:0: channel status CC bits 6:0
5. Pin widget: SET_CONVERTER_CHANNEL_TO_DIGITAL_SLOT_MAPPING= 0x22 and SET_CONVERTER_CHANNEL_TO_DIGITAL_SLOT_MAPPING= 0x33 to undo default swap of channels 2 and 3
6. AOC widget: SET_STRIPE_CONTROL to select number of lanes. Recommend programming to 2 lanes all the time.
7. Pin widget: SET_PIN_WIDGET_CTRL = 0x40, to enable output and use PCM packets.
8. If desired to use HDA for info packets, program them with SET_DIP_INDEX/SET_DIP_DATA and enable with SET_DIP_XMIT_CTRL.

23.2.11 HDMI Audio

Two methods of generating the CTS portion of ACR packets are supported:

- In “Force SW CTS”, software computes CTS for the current audio and pixel clocks and writes it to registers.
- In “Hardware CTS”, hardware computes CTS automatically.

Likewise, two methods for generating N portion are supported.

- In “SW N”, software programs current desired N into 0441_SUBPACK_HIGH register and AUDIO_N.VALUE.
- In “HW N”, software programs all the ACR_*_SUBPACK_HIGH per-frequency N, then hardware chooses the one to use.

1. Set NV_PDISP_HDMI_ACR_CTRL (all fields) to 0
2. Reset N counter: HDMI.NV_PDISP_AUDIO_N.RESETF = ASSERT, GENERATE=ALTERNATE, LOOKUP per table
3. Based on pixel clock and audio sample clock frequencies, program the appropriate ACR N value into NV_PDISP_HDMI_ACR_0441_SUBPACK_HIGH (see HDMI 1.2a specification, section 7.2.3). LSB of N goes in SB6, and MSB into SB4.
4. Likewise, program the appropriate ACR CTS value into NV_PDISP_HDMI_ACR_0441_SUBPACK_LOW. The LSB of CTS goes in SB3, and the MSB into SB1.

5. Program the computed AVAL into the appropriate SOR_AUDIO_AVAL_ register
6. Set NV_PDISP_AUDIO_N.VALUE = N – 1 (from above)
7. Set NV_PDISP_HDMI_ACR_*_SUBPACK_HIGH.ENABLE = YES.
8. Set NV_PDISP_HDMI_SPARE = 0x10000 OR'd with bits 0 and 1
9. Bring the N counter of reset: set HDMI.NV_PDISP_AUDIO_N.RESETF = DEASSERT
10. Program the audio source with SOR_AUDIO_CNTRL0.SOURCE_SELECT = HDAL for HDA, or AUTO. AUTO automatically selects HDA if HDMI's internal HDA codec has been enumerated by HDA controller.
11. 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.

23.2.12 Other HDMI Configurations

1. Set HDMI_CTRL.MAX_AC_PACKET based on horizontal blanking width. Using display timing registers, MAX_AC_PACKET = floor((H_SYNC_WIDTH + H_BACK_PORCH + H_FRONT_PORCH – HDMI_CTRL.REKEY – 18) / 32)
2. Program and enable Audio InfoFrames. When using HDA audio, its contents come from verbs sent to the internal codec.
3. Program and enable AVI InfoFrames. When using HDA Audio, this *may* come from the HDA Codec if desired, using one of its Generic packets. If so, do not enable the HDMI one:
   - Set HDMI_AVI_INFOFRAME_HEADER = 0x000D0282
   - Program checksum, PB1, PB2, PB3 in HDMI_AVI_INFOFRAME_SUBPACK0_LOW. PB4 contains the video format code from the CEA-861-D specification. See HDMI and EIA/CEA-861-D for all the fields. CRC is computed over header and PB1…PB13 as above.
   - Program PB4…PB6 in HDMI_AVI_INFOFRAME_SUBPACK0_HIGH, and PB7..PB13 in HDMI_AVI_INFOFRAME_SUBPACK1_LOW / HIGH.
   - Set HDMI_AVI_INFOFRAME_CTRL.ENABLE = EN
4. For HDMI 1.4 3D, program and enable the GENERIC InfoFrame to send an HDMI Vendor-Specific InfoFrame. When using HDA Audio, this *might* come from the HDA Codec, if desired, using one of its Generic packets. If so, do not enable the HDMI one.
   - Set HDMI_GENERIC_HEADER = 0x00LL0181 where LL is length of packet – 0x05 for the basic version described here.
   - HDMI_GENERIC_SUBPACK0_LOW.PB1/PB2/PB3 = 0x03/0c/00 (IEEE registration ID)
   - PB4 = 0x40 (Hdmi_video_format = 3d_structure present)
   - PB5 = 0x00 (3D_Structure = Frame packing)
   - PB6...PB27 = 0 (no 3D_Ext_Data for Frame packing)
   - Compute checksum over header and PB1..PB27 and put in PB0, as above.
   - HDMI_GENERIC_CTRL.ENABLE=EN. OTHER, SINGLE, HBLANK= DIS.
   - Be sure to keep HDMI_GENERIC_CTRL.AUDIO = EN

23.2.13 Start HDMI SOR

1. Enable TMDS macro with SOR_PLL0.PWR=0, VCOPD=0, PULLDOWN=0. Observe the power sequencing requirement. After PWR is cleared, you must wait 10 microseconds before setting PDBG=0.
2. Write SOR_PWR with NORMAL_STATE = PU, SAFE_STATE = PD, SETTING_NEW = TRIGGER.
3. Write SOR_PWR with the same settings but with SETTING_NEW = DONE.
4. Poll SOR_PWR for SETTING_NEW = DONE.
5. Write SOR_STATE2 with ASY_OWNER=HEAD0, SUBOWNER=3, ASY_PROTOCOL=SINGLE_TMDS_A, ASY_HSYNCPOL=POSITIVE_TRUE, ASY_VSYNCPOL=POSITIVE_TRUE, ASY_DEPOL=POSITIVE_TRUE.
6. Write SOR_STATE1 with ASY_HEAD_OPMODE=AWAKE, ASY_ORMODE=NORMAL, ATTACHED=0, ARM_SHOW_VGA=0.
7. Write SOR_STATE0 with UPDATE=0.
8. Write SOR_STATE0 with UPDATE=1.
9. Write SOR_STATE1 with ATTACHED=1.
10. Write SOR_STATE0 with UPDATE=0.

23.2.14 Start Display

- Set DISP_WIN_OPTIONS.HDMI_ENABLE = ENABLE.
- Start the display with DISPLAY_COMMAND.DISPLAY_CTRL_MODE and DISPLAY_POWER_CONTROL.

23.3 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 specifications.

23.3.1 Hot-Plug and ELD (EDID-like data)

When the HDMI driver receives notification of a hot-plug interrupt, it reads the downstream TV's E-EDID ROM. The EDID contains audio-related data, so the audio driver needs it, too. The HDMI can send the hot-plug event and EDID buffer to the Audio driver via the HDA Codec.

Note:	See register comments in SOR_AUDIO_HDA_PRESENSE and SOR_AUDIO_HDA_ELD_BUFWR later in this section.

23.3.2 Copy-Protection

The audio driver can query copy-protection state (i.e., HDCP enabled), and request that HDCP be enabled by the HDMI driver.

Note:	See SOR_AUDIO_HDA_CP register and CP_REQUEST interrupt in INT_STATUS later in this section.

23.3.3 InfoPacket

In HDA mode, the Audio InfoFrame data is provided by the internal HDA Codec, not by the HDMI_AUDIO_INFOFRAME* registers. The SET_DIP_INDEX/SET_DIP_DATA verbs write this data.

Three generic infopackets are provided by the HDA Codec, in addition to the exiting generic packet.

23.3.4 Miscellaneous

For Audio to HDMI signaling, the HDA Codec has SCRATCH verbs that are reflected in the HDMI SOR_AUDIO_HDA_CODEC_SCRATCH* registers and CODEC_SCRATCH* interrupts.

For HDMI to Audio signaling, HDMI SOR_AUDIO_HDA_SCRATCH* registers, when written, are reflected to the HDA Codec GET_NV_SCRATCH_REGN_FROM_DISP verb and cause an Unsolicited Response (interrupt in the HDA Controller).

23.4 Clock Use Cases

Tegra K1 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

23.5 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));

}
```

23.6 HDMI Registers

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

23.6.1 HDMI_CTXSW_0

Channel IDs

Context switch register. Should be common to all modules. Includes the current channel/class (which is writable by software) and the next channel/class (which the hardware sets when it receives a context switch).

Context switch works like this: Any context switch request triggers an interrupt to the host and causes the new channel/class to be stored in NEXT_CHANNEL/NEXT_CLASS (see vmod/chexample). Software sees that there is a context switch interrupt and does the necessary operations to make the module ready to receive traffic from the new context. It clears the context switch interrupt and writes CURR_CHANNEL/CLASS to the same value as NEXT_CHANNEL/CLASS, which causes a context switch acknowledge packet to be sent to the host. This completes the context switch and allows the host to continue sending data to the module.

Context switches can also be pre-loaded. If CURR_CLASS/CHANNEL are written and updated to the next CLASS/CHANNEL before the context switch request occurs, an acknowledge will be generated by the module and no interrupt will be triggered. This is one way for software to avoid dealing with context switch interrupts. Another way to avoid context switch interrupts is to set the AUTO_ACK bit.

This bit tells the module to automatically acknowledge any incoming context switch requests without triggering an interrupt. CURR_* and NEXT_* will be updated by the module so they will always be current.

Offset: 0x0 │ Byte Offset: 0x0 │ Read/Write: R/W │ Reset: 0xXXXXf800 (0bxxxxxxxxxxxxxxxx11111x0000000000)

Bit	R/W	Reset	Description
31:28	RO	X	NEXT_CHANNEL: Next requested channel
25:16	RO	X	NEXT_CLASS: Next requested class
15:12	RW	0xf	CURR_CHANNEL: Current working channel, reset to 'invalid'
11	RW	0x1	AUTO_ACK: Automatically acknowledge any incoming context switch requests 0 = MANUAL 1 = AUTOACK
9:0	RW	0x0	CURR_CLASS: Current working class

23.6.2 HDMI_NV_PDISP_SOR_STATE0_0

Writing a 1 to this field causes a 1-cycle pulse, which is used to activate the fields in registers NV_PDISP_SOR_STATE[1,2].

The ATTACHED field in SOR_STATE1 is flopped to create the read-only field "ATTACHED" in the NV_PDISP_SOR_TEST register.

Offset: 0x1 | Byte Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	UPDATE

23.6.3 HDMI_NV_PDISP_SOR_STATE1_0

Offset: 0x2 | Byte Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	ARM_SHOW_VGA
3	0x0	ATTACHED
2	SAFE	ASY_ORMODE: 0 = SAFE 1 = NORMAL
1:0	0x0	ASY_HEAD_OPMODE: 0 = SLEEP 1 = SNOOZE 2 = AWAKE

23.6.4 HDMI_NV_PDISP_SOR_STATE2_0

Offset: 0x3 | Byte Offset: 0xc | Read/Write: R/W | Reset: 0x00000151 (0bxxxxxxxxxxxxxxxxx000000101010001) | Default: 0x00000030

Bit	Reset	Default	Description
14	0x0	NONE	ASY_DEPOL: 0 = POSITIVE_TRUE 1 = NEGATIVE_TRUE
13	0x0	NONE	ASY_VSYNCPOL: VSYNCPOL depends on the video mode. 0 = POSITIVE_TRUE 1 = NEGATIVE_TRUE
12	0x0	NONE	ASY_HSYNCPOL: HSYNCPOL depends on the video mode. There is no single correct value. There are unlimited modes. 0 = POSITIVE_TRUE 1 = NEGATIVE_TRUE
11:8	SINGLE_TMDS_A	NONE	ASY_PROTOCOL: 1 = SINGLE_TMDS_A 15 = CUSTOM
7:6	COMPLETE_RASTER	NONE	ASY_CRCMODE: 0 = ACTIVE_RASTER 1 = COMPLETE_RASTER 2 = NON_ACTIVE_RASTER
5:4	SUBHEAD0	BOTH	ASY_SUBOWNER: 0 = NONE 1 = SUBHEAD0 2 = SUBHEAD1 3 = BOTH
3:0	HEAD0	NONE	ASY_OWNER: 0 = NONE 1 = HEAD0

23.6.5 HDMI_NV_PDISP_HDMI_AUDIO_EMU0_0

HDMI_AUDIO_EMU

HDMI_AUDIO_EMU0 and HDMI_AUDIO_EMU1 are used to program the S/PDIF transmitter during RTL simulation and emulation. This is strictly for verification. This register is write-only.

Offset: 0x1a │ Byte Offset: 0x68 │ Read/Write: WO │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	REGX

23.6.6 HDMI_NV_PDISP_HDMI_AUDIO_EMU_RDATA0_0

HDMI_AUDIO_EMU_RDATA0 is read-only.

Offset: 0x1b │ Byte Offset: 0x6c │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RDATA

23.6.7 HDMI_NV_PDISP_HDMI_AUDIO_EMU1_0

HDMI_AUDIO_EMU1 and EMU2 allow control over S/PDIF transmitter clock ratios (which are fixed in the GPU). They do indirect addressing.

EMU1 has the address, and EMU2 has the data.

To write:

1. Write data register with data

2. Write address register with bit 31=1, and address in 15:0

3. Write address register with bit 31=0

To read:

1. Write address register with bit 31=0 and address in 15:0

2. Read back address register to introduce delay allowing read data to propagate

3. Read data register

Offset: 0x1c │ Byte Offset: 0x70 │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31	0x0	WRITE
7:0	0x0	ADDR: 0 = CNT_22 1 = CNT_24 2 = CNT_32 3 = CNT_44 4 = CNT_48 5 = CNT_88 6 = CNT_96 7 = CNT_176 8 = CNT_192 9 = JIT_05_22 10 = JIT_05_24 11 = JIT_05_32 12 = JIT_05_44 13 = JIT_05_48 14 = JIT_05_88 15 = JIT_05_96

Bit	Reset	Description
		16 = JIT_05_176 17 = JIT_05_192 18 = JIT_10_22 19 = JIT_10_24 20 = JIT_10_32 21 = JIT_10_44 22 = JIT_10_48 23 = JIT_10_88 24 = JIT_10_96 25 = JIT_10_176 26 = JIT_10_192 27 = JIT_11_22 28 = JIT_11_24 29 = JIT_11_32 30 = JIT_11_44 31 = JIT_11_48 32 = JIT_11_88 33 = JIT_11_96 34 = JIT_11_176 35 = JIT_11_192 36 = GLITCH_PERIOD_22 37 = GLITCH_PERIOD_24 38 = GLITCH_PERIOD_32 39 = GLITCH_PERIOD_44 40 = GLITCH_PERIOD_48 41 = GLITCH_PERIOD_88 42 = GLITCH_PERIOD_96 43 = GLITCH_PERIOD_176 44 = GLITCH_PERIOD_192 45 = GLITCH_CNT_22 46 = GLITCH_CNT_24 47 = GLITCH_CNT_32 48 = GLITCH_CNT_44 49 = GLITCH_CNT_48 50 = GLITCH_CNT_88 51 = GLITCH_CNT_96 52 = GLITCH_CNT_176 53 = GLITCH_CNT_192

23.6.8 HDMI_NV_PDISP_HDMI_AUDIO_EMU2_0

Offset: 0x1d │ Byte Offset: 0x74 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA

23.6.9 HDMI_NV_PDISP_HDMI_AUDIO_INFOFRAME_CTRL_0

This is the register space for the HDMI block. Refer to HDMI Version 1.1. Refer to IEC60958-1 for general information and IEC60958-3 for consumer specifications for related audio information.

Additional notes regarding the ACR packet:

There are several methods that can be used to generate the N and CTS values in the Audio Clock Regeneration packet. The details of these methods are described here.

Full Software control: Software controls the contents of the possible ACR packets. Software writes the contents of the ACR_XXXX_SUBPACK_HIGH and ACR_XXXX_SUBPACK_LOW for all seven audio frequencies.

Software selects a method in ACR_CTRL to determine the audio sampling frequency

1. Measured in the audio block (MEASURE)
2. Read from the channel status information encoded in the audio stream (PACKET)

3. Defined by software (FREQS)

Software writes _YES to the ENABLE field of the SUBPACK_HIGH registers to allow that pair of registers to be used as the ACR packet. The audio sampling frequency specified in ACR_CTRL will select one of the seven pairs of SUBPACK registers to use as the ACR packet. This packet is sent on every Frame 27 of the audio block if that pair's ENABLE field is _YES.

Hardware controlled CTS: Software provides the N value and the hardware measures the CTS value. This is described in the HDMI_SPARE comments and mentioned in all other affected registers.

Software writes _ENABLE to HDMI_SPARE_HW_CTS to enable this feature.

Software writes 1 to HDMI_SPARE_CTS_RESET_VAL. This controls what the internal CTS counter resets to when it starts the next round of measurement. Resetting to zero was not correct, and adding a few bits of control helped to fine tune the measurement.

Software sets all ENABLEs in ACR_CTRL to _NO. (The easy way to do this is just to write zero to the whole register.)

Software writes the value of N in two places:

1. ACR_0441_SUBPACK_HIGH_SB[4/5/6]: It is written here so that the ACR packet contains the correct information for N.

2. AUDIO_N_VALUE: It is written here so that the audio block knows what N to use while it is measuring CTS. (Software does not need to write N to these places if Hardware Selected N is used.)

Software enables the sending of ACR_0441_SUBPACK* by setting ACR_0441_SUBPACK_ENABLE to _YES. (This pair of registers is used for all audio frequencies when Hardware controlled CTS is enabled.)

Software writes _USE_HW_CTS_VAL to ACR_0441_SUBPACK_LOW_SB1.

Hardware selected N: Software provides the possible N values whenever the pixel clock frequency changes. Hardware looks up the correct N value from a table stored in priv registers.

Software writes the 7 possible N values into the 7 AUDIO_NVAL registers. This is based on pixel clock frequency.

Software enables this feature by setting AUDIO_N_LOOKUP to _ENABLE. Hardware will use the audio sampling frequency detected by the audio block to select the correct N value.

NOTE: This is not the audio sampling frequency reported by the channel status information in the audio stream. Hardware will ignore whatever if written in the SUBPACK_HIGH registers when this feature is enabled and send the value of N it selected in these bytes of the ACR packet. CTS can come from the SUBPACK_LOW registers or the Hardware controlled CTS.

Audio frequency notes: There are several places in the HDMI and Audio registers where the audio sampling frequency is reported or set. Here is a breakdown of all such fields.

In SPDIF audio encoding, one bit of each audio sample is part of the "channel status" information. The bits from all 192 audio samples combine to form the 192 bit channel status packet. For consumer applications, only the first 40 bits have defined values. These bits are included in the encoded HDMI audio packet. Four of these bits contain the audio sampling frequency as reported by the audio source. See the IEC60958-3 specification for more information.

SPDIF_CHN_STATUS1 and SPDIF_CHN_STATUS2 are read-only registers that contain the 40 bits of channel status data read from the last audio block. The audio frequency the audio stream reports can be read from SPDIF_CHN_STATUS1_SFREQ. This is the sampling frequency that ACR_CTRL refers to with _PACKET.

SPDIF_CHN_STATUS1_ORIGINAL is the original sampling frequency of the audio. If the source of the audio stream changed the sampling frequency from its original frequency for any reason, the original sampling frequency is reported here.

AUDIO_CNTRL0_SAMPLING_FREQ is the sampling frequency measured by the audio block. It counts the number of dispclks periods that occur between to specific points in the audio stream. It compares this count to the thresholds in the AUDIO_FS registers to determine what the audio sampling frequency is. This is the sampling frequency that ACR_CTRL refers to as

_MEASURE. This is the sampling frequency used to determine the correct value of N when Hardware selected N is enabled (see AUDIO_N).

The CHANNEL_STATUS1 and CHANNEL_STATUS2 registers are for debug. They can be used to replace the channel status bits that were in the original SPDIF stream with user-defined bits. CHANNEL_STATUS1_SFREQ can be used to report a different sampling frequency to downstream devices.

ACR_CTRL_FREQS can be written to select a specific pair of SUBPACK registers to send as your ACR packet. ACR_CTRL_FREQS_ENABLE must be set to _YES for this to have any effect.

The HDMI Audio InfoFrame, AVI InfoFrame, and Generic InfoFrame have nearly identical priv register interfaces. The next five registers control the Audio InfoFrame as described in section 8.2.2 of the HDMI specification.

HDMI_AUDIO_INFOFRAME_CTRL

This register controls the frequency and generation of audio InfoFrame packets. The contents of the packet should be written into the header (HDMI_AUDIO_INFOFRAME_HEADER) and subpacket (HDMI_AUDIO_INFOFRAME_SUBPACK0_LOW, HDMI_AUDIO_INFOFRAME_SUBPACK0_HIGH) registers.

ENABLE: Setting this field to _YES will initiate InfoFrame generation. Setting this bit to _NO will disable InfoFrame generation at the beginning of the next frame. The frequency of InfoFrame generation is controlled by OTHER and SINGLE fields.

OTHER: Setting this field to _EN while SINGLE is set to _DIS will cause InfoFrame to be transmitted to every other frame.

SINGLE: Setting this field to _EN while OTHER is set to _DIS will cause InfoFrame to be transmitted exactly once.

If OTHER and SINGLE fields are both set to _DIS, an InfoFrame will be generated every frame. Software should never set OTHER and SINGLE both to _EN.

See chapter 8.2.2 - Audio InfoFrame, in the HDMI specification for more information

CHKSUM_HW: Hardware provides a way to calculate the checksum for the InfoFrames.

_ENABLE will enable the hardware calculation to be passed to the packet

_DISABLE:

- Uses the register value defined in NV_PDISP_SOR_HDMI_AUDIO_INFOFRAME_SUBPACK0_LOW_PB0 for the S/PDIF
- Uses the checksum provided by the Azalia codec for the Azalia audio formats.

Usage: Normal Operation

Offset: 0x1e │ Byte Offset: 0x78 │ Read/Write: R/W │ Reset: 0x00000200 (0bxxxxxxxxxxxxxxxxxxxxxxx10xxx0xxx0)

Bit	Reset	Description
9	ENABLE	CHKSUM_HW: CHKSUM_HW is an optional feature for software to use. Not required to be enabled. 0 = DISABLE 1 = ENABLE
8	0x0	SINGLE: 0 = DIS 1 = EN
4	0x0	OTHER: 0 = DIS 1 = EN

Bit	Reset	Description
0	0x0	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

23.6.10 HDMI_NV_PDISP_HDMI_AUDIO_INFOFRAME_STATUS_0

HDMI_AUDIO_INFOFRAME_STATUS

The SENT bit will be set to _DONE, after the first packet is sent. After the ENABLE bit in INFOFRAME_CTRL is set to _NO, the SENT bit will be set to _WAITING after the start of the next frame to indicate that it is safe to change the contents of the packet header and subpacket registers.

Usage: Normal Operation

Offset: 0x1f │ Byte Offset: 0x7c │ Read/Write: RO │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	SENT: 1 = DONE 0 = WAITING

23.6.11 HDMI_NV_PDISP_HDMI_AUDIO_INFOFRAME_HEADER_0

HDMI_AUDIO_INFOFRAME_HEADER

This register should be written with the value of the HDMI Audio InfoFrame header. This header is described in table 8-4 of chapter 8.2.2 of the HDMI specification. HB0, HB1, and HB2 are defined fields of the header from the specification.

Offset: 0x20 │ Byte Offset: 0x80 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:16	0x0	HB2
15:8	0x0	HB1
7:0	0x0	HB0

23.6.12 HDMI_NV_PDISP_HDMI_AUDIO_INFOFRAME_SUBPACK0_LOW_0

HDMI_AUDIO_INFOFRAME_SUBPACK0_LOW

This register should be written with the lower 4 bytes of the HDMI Audio InfoFrame. PB0, PB1, PB2, and PB3 are defined fields from the specification. See chapter 8.2.2 - Audio InfoFrame, Table 8-5, in the HDMI specification for more information

Offset: 0x21 │ Byte Offset: 0x84 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB3
23:16	0x0	PB2
15:8	0x0	PB1
7:0	0x0	PB0

23.6.13 HDMI_NV_PDISP_HDMI_AUDIO_INFOFRAME_SUBPACK0_HIGH_0

HDMI_AUDIO_INFOFRAME_SUBPACK0_HIGH

This register should be written with the upper 2 bytes of the HDMI Audio InfoFrame. PB4 and PB5 are defined fields from the specification. See chapter 8.2.2 - Audio InfoFrame, Table 8-5, in the HDMI specification for more information

Offset: 0x22 │ Byte Offset: 0x88 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PB5
7:0	0x0	PB4

23.6.14 HDMI_NV_PDISP_HDMI_AVI_INFOFRAME_CTRL_0

The following registers are used to send a 14-byte packet intended for sending AVI InfoFrame packet as described in section 8.2.1 of the HDMI specification.

HDMI_AVI_INFOFRAME_CTRL

This register controls the frequency and generation of AVI InfoFrame packets. The fields of this register are identical to HDMI_AUDIO_INFOFRAME_CTRL. See chapter 8.2.1 - Auxiliary Video information (AVI) InfoFrame, in the HDMI specification for more information

Offset: 0x23 │ Byte Offset: 0x8c │ Read/Write: R/W │ Reset: 0x00000200 (0bxxxxxxxxxxxxxxxxxxxxxxxx10xxx0xxx0)

Bit	Reset	Description
9	ENABLE	CHKSUM_HW: CHKSUM_HW is an optional feature for software to use. Not required to be enabled 0 = DISABLE 1 = ENABLE
8	DIS	SINGLE: 0 = DIS 1 = EN
4	DIS	OTHER: 0 = DIS 1 = EN
0	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

23.6.15 HDMI_NV_PDISP_HDMI_AVI_INFOFRAME_STATUS_0

HDMI_AVI_INFOFRAME_STATUS

The SENT bit will be set to _DONE, after the first packet is sent. After the ENABLE bit in INFOFRAME_CTRL is set to _NO, the SENT bit will be set to _WAITING after the start of the next frame to indicate that it is safe to change the contents of the packet header and subpacket registers.

Offset: 0x24 │ Byte Offset: 0x90 │ Read/Write: RO │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	SENT: 1 = DONE 0 = WAITING

23.6.16 HDMI_NV_PDISP_HDMI_AVI_INFOFRAME_HEADER_0

HDMI_AVI_INFOFRAME_HEADER

This register should be written with the value of the HDMI AVI InfoFrame header. This header is described in table 8-1 of chapter 8.2.1 of the HDMI specification. HB0, HB1, and HB2 are defined fields of the header from the specification.

Offset: 0x25 │ Byte Offset: 0x94 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:16	0x0	HB2
15:8	0x0	HB1
7:0	0x0	HB0

23.6.17 HDMI_NV_PDISP_HDMI_AVI_INFOFRAME_SUBPACK0_LOW_0

HDMI_AVI_INFOFRAME_SUBPACK0_LOW

This register should be written with the lower 4 bytes of the HDMI AVI InfoFrame. PB0, PB1, PB2, and PB3 are defined fields from the specification. See chapter 8.2.1 - Auxiliary Video information (AVI) InfoFrame, Table 8-2, in the HDMI specification for more information.

Offset: 0x26 │ Byte Offset: 0x98 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB3
23:16	0x0	PB2
15:8	0x0	PB1
7:0	0x0	PB0

23.6.18 HDMI_NV_PDISP_HDMI_AVI_INFOFRAME_SUBPACK0_HIGH_0

HDMI_AVI_INFOFRAME_SUBPACK0_HIGH

This register should be written with bytes 4-6 of the HDMI AVI InfoFrame. PB3, PB5, and PB6 are defined fields from the specification. See chapter 8.2.1 - Auxiliary Video information (AVI) InfoFrame, Table 8-2, in the HDMI specification for more information.

Offset: 0x27 │ Byte Offset: 0x9c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB6
15:8	0x0	PB5
7:0	0x0	PB4

23.6.19 HDMI_NV_PDISP_HDMI_AVI_INFOFRAME_SUBPACK1_LOW_0

HDMI_AVI_INFOFRAME_SUBPACK1_LOW

This register should be written with bytes 7-10 of the HDMI AVI InfoFrame. PB7, PB8, PB9, and PB10 are defined fields from the specification. See chapter 8.2.1 - Auxiliary Video information (AVI) InfoFrame, Table 8-2, in the HDMI specification for more information.

Offset: 0x28 │ Byte Offset: 0xa0 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB10
23:16	0x0	PB9
15:8	0x0	PB8
7:0	0x0	PB7

23.6.20 HDMI_NV_PDISP_HDMI_AVI_INFOFRAME_SUBPACK1_HIGH_0

HDMI_AVI_INFOFRAME_SUBPACK1_HIGH

This register should be written with bytes 11-13 of the HDMI AVI InfoFrame. PB11, PB12, and PB13 are defined fields from the specification. See chapter 8.2.1 - Auxiliary Video information (AVI) InfoFrame, Table 8-2, in the HDMI specification for more information.

Offset: 0x29 │ Byte Offset: 0xa4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB13
15:8	0x0	PB12
7:0	0x0	PB11

23.6.21 HDMI_NV_PDISP_HDMI_GENERIC_CTRL_0

The following registers are used to send a 28-byte packet intended for sending any packet type. These registers will most likely be used for debug purposes.

HDMI_GENERIC_CTRL

This register controls the frequency and generation of generic InfoFrame packets. The fields of this register are identical to HDMI_AUDIO_INFOFRAME_CTRL except where noted below.

HBLANK: If HBLANK is set to _EN, then this packet will be sent (once) during the next horizontal blanking interval. The packet will still be sent at most once per frame. Software can poll GENERIC_STATUS_SENT to determine when the packet has been sent.

Using HBLANK is intended to mimic Audio Sample Packets and ACR Packets and therefore sending of the actual audio packets should be disabled by setting AUDIO to _DIS. This feature is used for debug purposes only.

To mimic an audio packet:

- begin
- wait for GENERIC_STATUS_SENT=_WAITING
- write next 4 audio samples to GENERIC_HEADER/GENERIC_SUBPACK registers
- set HBLANK = _EN
- wait for GENERIC_STATUS_SENT=_SENT
- set HBLANK = _DIS
- end

AUDIO: Audio packet transmission will be stopped when set to _DIS. Normal audio packet transmission will be allowed when set to _EN. This is used during debug to disable normal audio when using HBLANK.

Offset: 0x2a │ Byte Offset: 0xa8 │ Read/Write: R/W │ Reset: 0x00010000 (0bxxxxxxxxxxxxxxxx1xxx0xxx0xxx0xxx0)

Bit	Reset	Description
16	EN	AUDIO: Set to 1 in audio mode and 0 in DVI mode. 0 = DIS 1 = EN
12	DIS	HBLANK: 0 = DIS 1 = EN
8	DIS	SINGLE: 0 = DIS 1 = EN
4	DIS	OTHER: 0 = DIS 1 = EN
0	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

23.6.22 HDMI_NV_PDISP_HDMI_GENERIC_STATUS_0

HDMI_GENERIC_STATUS

The SENT bit will be set to _DONE, after the first packet is sent. After the ENABLE bit in INFOFRAME_CTRL is set to _NO, the SENT bit will be set to _WAITING after the start of the next frame to indicate that it is safe to change the contents of the packet header and subpacket registers.

Offset: 0x2b │ Byte Offset: 0xac │ Read/Write: RO │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	SENT: 1 = DONE 0 = WAITING

23.6.23 HDMI_NV_PDISP_HDMI_GENERIC_HEADER_0

HDMI_GENERIC_HEADER

This register should be written with the contents of the packet header.

Offset: 0x2c │ Byte Offset: 0xb0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	HB2
15:8	0x0	HB1
7:0	0x0	HB0

23.6.24 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK0_LOW_0

HDMI_GENERIC_SUBPACK0_LOW

Bytes 0-3 of the packet are written into this register.

Offset: 0x2d │ Byte Offset: 0xb4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB3
23:16	0x0	PB2
15:8	0x0	PB1
7:0	0x0	PB0

23.6.25 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK0_HIGH_0

HDMI_GENERIC_SUBPACK0_HIGH

Bytes 4-6 of the packet are written into this register.

Offset: 0x2e │ Byte Offset: 0xb8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB6
15:8	0x0	PB5

Bit	Reset	Description
7:0	0x0	PB4

23.6.26 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK1_LOW_0

HDMI_GENERIC_SUBPACK1_LOW

Bytes 7-10 of the packet are written into this register.

Offset: 0x2f │ Byte Offset: 0xbc │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB10
23:16	0x0	PB9
15:8	0x0	PB8
7:0	0x0	PB7

23.6.27 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK1_HIGH_0

HDMI_GENERIC_SUBPACK1_HIGH

Bytes 11-13 of the packet are written into this register.

Offset: 0x30 │ Byte Offset: 0xc0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB13
15:8	0x0	PB12
7:0	0x0	PB11

23.6.28 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK2_LOW_0

HDMI_GENERIC_SUBPACK2_LOW

Bytes 14-17 of the packet are written into this register.

Offset: 0x31 │ Byte Offset: 0xc4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB17
23:16	0x0	PB16
15:8	0x0	PB15
7:0	0x0	PB14

23.6.29 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK2_HIGH_0

HDMI_GENERIC_SUBPACK2_HIGH

Bytes 18-20 of the packet are written into this register.

Offset: 0x32 │ Byte Offset: 0xc8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB20
15:8	0x0	PB19
7:0	0x0	PB18

23.6.30 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK3_LOW_0

HDMI_GENERIC_SUBPACK3_LOW

Bytes 21-24 of the packet are written into this register.

Offset: 0x33 │ Byte Offset: 0xcc │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB24
23:16	0x0	PB23
15:8	0x0	PB22
7:0	0x0	PB21

23.6.31 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK3_HIGH_0

HDMI_GENERIC_SUBPACK3_HIGH

Bytes 25-27 of the packet are written into this register.

Offset: 0x34 │ Byte Offset: 0xd0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB27
15:8	0x0	PB26
7:0	0x0	PB25

23.6.32 HDMI_NV_PDISP_HDMI_ACR_CTRL_0

HDMI_ACR_CTRL

The Audio Clock Regeneration (ACR) packet contains the N and CTS values that the HDMI sink requires to recreate the audio clock. This mechanism is described in chapter 7.2 of the HDMI specification.

If software needs to specify N and CTS directly, this register is used to select the audio sampling frequency detection mechanism. The sampling rate is used to index one of the seven ACR_XXXX_SUBPACK_LOW/HIGH registers which must be written with the correct value of N and CTS. N and CTS are determined by the current audio sampling frequency and the pixel clock frequency:

CTS = (PixelClock*N) / (128*AudioSamplingFrequency)

The selected sampling rate must point to a valid entry (i.e., one of the 7 listed below) and the selected ACR_XXXX_SUBPACK_HIGH_ENABLE must be set to enable sending the packet.

When software is controlling N and CTS directly, the ACR packet is sent every 27th audio frame (of 0 to 191) of the audio block. The sampling frequency read from the channel status bits is not available until the 27th audio frame.

PACKET_ENABLE: Set this to _YES to use the channel status information read from the incoming SPDIF audio stream to determine the sampling frequency.

MEASURE_ENABLE: Set this to _YES to use the sampling frequency measured in the in the audio block to determine the sampling frequency. This is the sampling frequency that is read from AUDIO_CNTRL0_SAMPLING_FREQ.

FREQS_ENABLE: Set this to _YES to use the sampling frequency written into the _FREQS field of this register.

FREQS: This is the audio sampling frequency that will be used when FREQS_ENABLE is _YES.

When Hardware is used to measure CTS, all ENABLE fields of this register should be set to _NO.

All four subpackets contain the same ACR packet.

Offset: 0x35 │ Byte Offset: 0xd4 │ Read/Write: R/W │ Reset: 0x02010000 (0bxxxx0010xxxxxxx1xxxxxxx0xxxxxxx0)

Bit	Reset	Description
27:24	FREQ_48KHZ	FREQS: 3 = FREQ_32KHZ 0 = FREQ_44_1KHZ 2 = FREQ_48KHZ 8 = FREQ_88_2KHZ 10 = FREQ_96KHZ 12 = FREQ_176_4KHZ 14 = FREQ_192KHZ
16	YES	FREQS_ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
8	NO	MEASURE_ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
0	NO	PACKET_ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

23.6.33 HDMI_NV_PDISP_HDMI_ACR_0320_SUBPACK_LOW_0

HDMI_ACR_0320_SUBPACK_LOW

Contains bytes 1-3 of the 32 kHz ACR packet. This is the CTS value.

See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

Offset: 0x36 │ Byte Offset: 0xd8 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000xxxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

23.6.34 HDMI_NV_PDISP_HDMI_ACR_0320_SUBPACK_HIGH_0

HDMI_ACR_0320_SUBPACK_HIGH

Contains bytes 4-6 of the 32 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

ENABLE: _YES allows this packet to be sent. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0x37 │ Byte Offset: 0xdc │ Read/Write: R/W │ Reset: 0x00000000 (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

23.6.35 HDMI_NV_PDISP_HDMI_ACR_0441_SUBPACK_LOW_0

HDMI_ACR_0441_SUBPACK_LOW

Contains bytes 1-3 of the 44.1 kHz ACR packet. This is the CTS value. If Hardware measured CTS is enabled, SB1 should be set to zero. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

Offset: 0x38 │ Byte Offset: 0xe0 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000xxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1: 0 = USE_HW_CTS_VAL
23:16	0x0	SB2
15:8	0x0	SB3

23.6.36 HDMI_NV_PDISP_HDMI_ACR_0441_SUBPACK_HIGH_0

HDMI_ACR_0441_SUBPACK_HIGH

Contains bytes 4-6 of the 44.1 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

If Hardware measured CTS is enabled, ACR_0441_SUBPACK_HIGH_N should be written with the N value for the current audio sampling frequency.

If the hardware N value selection is enabled, the N value does not need to be written to this register.

ENABLE: _YES allows this packet to be sent. This should be set to _YES when hardware measured CTS is being used. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0x39 │ Byte Offset: 0xe4 │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
23:16	0x0	SB4
15:8	0x0	SB5
7:0	0x0	SB6

23.6.37 HDMI_NV_PDISP_HDMI_ACR_0882_SUBPACK_LOW_0

HDMI_ACR_0882_SUBPACK_LOW

Contains bytes 1-3 of the 88.2 kHz ACR packet. This is the CTS value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

Offset: 0x3a │ Byte Offset: 0xe8 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000xxxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

23.6.38 HDMI_NV_PDISP_HDMI_ACR_0882_SUBPACK_HIGH_0

HDMI_ACR_0882_SUBPACK_HIGH

Contains bytes 4-6 of the 88.2 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

ENABLE: _YES allows this packet to be sent. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0x3b │ Byte Offset: 0xec │ Read/Write: R/W │ Reset: 0x00000000 (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

23.6.39 HDMI_NV_PDISP_HDMI_ACR_1764_SUBPACK_LOW_0

HDMI_ACR_1764_SUBPACK_LOW

Contains bytes 1-3 of the 176.4 kHz ACR packet. This is the CTS value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

Offset: 0x3c │ Byte Offset: 0xf0 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000xxxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

23.6.40 HDMI_NV_PDISP_HDMI_ACR_1764_SUBPACK_HIGH_0

HDMI_ACR_1764_SUBPACK_HIGH

Contains bytes 4-6 of the 176.4 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

ENABLE: _YES allows this packet to be sent. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0x3d │ Byte Offset: 0xf4 │ Read/Write: R/W │ Reset: 0x00000000 (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

23.6.41 HDMI_NV_PDISP_HDMI_ACR_0480_SUBPACK_LOW_0

HDMI_ACR_0480_SUBPACK_LOW

Contains bytes 1-3 of the 48 kHz ACR packet. This is the CTS value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

Offset: 0x3e │ Byte Offset: 0xf8 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000xxxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

23.6.42 HDMI_NV_PDISP_HDMI_ACR_0480_SUBPACK_HIGH_0

HDMI_ACR_0480_SUBPACK_HIGH

Contains bytes 4-6 of the 48 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

ENABLE: _YES allows this packet to be sent. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0x3f │ Byte Offset: 0xfc │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
23:16	0x0	SB4
15:8	0x0	SB5
7:0	0x0	SB6

23.6.43 HDMI_NV_PDISP_HDMI_ACR_0960_SUBPACK_LOW_0

HDMI_ACR_0960_SUBPACK_LOW

Contains bytes 1-3 of the 96 kHz ACR packet. This is the CTS value. See table 5-11 in chapter 5.3.3 of the HDMI specification for more information.

Offset: 0x40 │ Byte Offset: 0x100 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000xxxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

23.6.44 HDMI_NV_PDISP_HDMI_ACR_0960_SUBPACK_HIGH_0

HDMI_ACR_0960_SUBPACK_HIGH

Contains bytes 4-6 of the 96 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

ENABLE: _YES allows this packet to be sent. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0x41 │ Byte Offset: 0x104 │ Read/Write: R/W │ Reset: 0x00000000 (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

23.6.45 HDMI_NV_PDISP_HDMI_ACR_1920_SUBPACK_LOW_0

HDMI_ACR_1920_SUBPACK_LOW

Contains bytes 1-3 of the 192 kHz ACR packet. This is the CTS value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

Offset: 0x42 │ Byte Offset: 0x108 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000xxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

23.6.46 HDMI_NV_PDISP_HDMI_ACR_1920_SUBPACK_HIGH_0

HDMI_ACR_1920_SUBPACK_HIGH

Contains bytes 4-6 of the 192 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

ENABLE: _YES allows this packet to be sent. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0x43 │ Byte Offset: 0x10c │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
23:16	0x0	SB4
15:8	0x0	SB5
7:0	0x0	SB6

23.6.47 HDMI_NV_PDISP_HDMI_CTRL_0

HDMI_CTRL

REKEY: REKEY is the number of clocks required for HDCP rekey, starting from when DE is deasserted. No HDMI packets can be sent during this time. Due to a two cycle delay in hardware, REKEY should be set to two less than the desired value.

AUDIO_LAYOUT: AUDIO_LAYOUT controls layout (HB1[4]) of the Audio Sample Packet Header. Two different layouts are supported in the HDMI specification. The hardware only supports layout _2CH (2 channel audio). This should not need to be programmed beyond its init value of _2CH.

See Section 5.3.4 of the HDMI specification.

AUDIO_LAYOUT_SELECT: For projects with integrated audio codec, the AUDIO_LAYOUT information is automatically detected by hardware (default). We can override this capability and fall back on AUDIO_LAYOUT based selection by setting AUDIO_LAYOUT_SELECT to _SW_BASED.

If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in the project specification, this field has no meaning, and only AUDIO_LAYOUT field will take effect.

SAMPLE_FLAT: SAMPLE_FLAT controls the values of (HB2[3:0]) of the Audio Sample Packet Header and should be _CLR. See Section 5.3.4 of the HDMI specification.

MAX_AC_PACKET: Set MAX_AC_PACKET to the maximum number of 32-pixel packets that will fit in the horizontal blanking interval. This controls the maximum number of audio packets, ACR packets, GCP packets, InfoFrames, etc. that will be sent during the horizontal blanking period.

MAX_AC_PACKET <= Floor[((HBLANK-REKEY-18)/32)]

CT_SELECT: Diagnostic workaround bit to decide whether the value of "coding type" will be hardware based or software based. The CT field is required in the construction of Audio InfoFrame packet (HDMI Specification, section 8.2.2). By default, the CT field is constructed by software. However, by setting this bit, the CT field in the construction of the Audio InfoFrame packet comes as a sideband signal from the HDA codec. If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in projects.spec, this field is reserved.

CC_SELECT: Diagnostic workaround bit to decide whether the value of "channel count" will be hardware based or software based. The CC field is required in the construction of Audio InfoFrame packet (HDMI Specification, section 8.2.2). By default, the CC field is constructed by software. However,by setting this bit, the CC field in the construction of the Audio InfoFrame packet comes as a sideband signal from the HDA codec. If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in projects.spec, this field is reserved.

SF_SELECT: Diagnostic workaround bit to decide whether the value of "sampling frequency" will be hardware based or software based. The SF field is required in the construction of Audio InfoFrame packet (HDMI Specification, section 8.2.2). By default, the SF field is constructed by software. However, by setting this bit, the SF field in the construction of the Audio InfoFrame packet comes as a sideband signal from the HDA codec. If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in projects.spec, this field is reserved.

SS_SELECT: Diagnostic workaround bit to decide whether the value of "sample size" will be hardware based or software based. The SS field is required in the construction of Audio InfoFrame packet (HDMI Specification, section 8.2.2). By default, the SS field is constructed by software. However, by setting this bit, the SS field in the construction of the Audio InfoFrame packet comes as a sideband signal from the HDA codec. If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in projects.spec, this field is reserved.

CA_SELECT: Diagnostic workaround bit to decide whether the value of "channel allocation" will be hardware based or software based. The CA field is required in the construction of Audio InfoFrame packet (HDMI Specification, section 8.2.2). By default, the CA field is constructed by software. However, by setting this bit, the CA field in the construction of the Audio InfoFrame packet comes as a sideband signal from the HDA codec. If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in projects.spec, this field is reserved.

ENABLE: Set to _YES to enable HDMI for this head. Set to _NO to disable HDMI for this head.

Offset: 0x44 │ Byte Offset: 0x110 │ Read/Write: R/W │ Reset: 0x00020038 (0bx0x00000xxx00010xxx0x0x0x0111000)

Bit	Reset	Description
30	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
28	SW	CA_SELECT: 0 = SW 1 = HW
27	SW	SS_SELECT: 0 = SW 1 = HW
26	SW	SF_SELECT:

Bit	Reset	Description
		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

23.6.48 HDMI_NV_PDISP_HDMI_VSYNC_KEEPOUT_0

HDMI_VSYNC_KEEPOUT

Defines the start and end of the VSYNC keepout period where HDMI packets should not be sent. This is defined in chapter 2.7 the HDCP 1.1 specification.

END: Defines the end of the keepout period. This is measured in pixels after the active edge of VSYNC. This is defined in the specification and should not need to be written beyond its initialized value.

START: Defines the start of the keepout period. This is measured in pixels after the active edge of VSYNC. This is defined in the specification and should not need to be written beyond its initialized value.

ENABLE: When set to _YES, the keepout window is respected and no HDMI packets are sent in the period of time between START and END. This bit should be set to _YES. When set to _NO, the keepout window is ignored and HDMI packets may be sent regardless of the keepout window.

Offset: 0x45 │ Byte Offset: 0x114 │ Read/Write: R/W │ Reset: 0x819a028a (0b1xxxxx0110011010xxxxxx1010001010)

Bit	Reset	Description
31	YES	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
25:16	0x19a	START
9:0	0x28a	END

23.6.49 HDMI_NV_PDISP_HDMI_VSYNC_WINDOW_0

HDMI_VSYNC_WINDOW

Defines the start and end of the window of opportunity where the HDCP EESS signaling occurs. This is defined in chapter 2.7 of the HDCP 1.1 specification.

END: Defines the end of the window of opportunity. This is measured in pixels after the active edge of VSYNC. This is defined in the specification and should not need to be written beyond its initialized value.

START: Defines the start of the window of opportunity. This is measured in pixels after the active edge of VSYNC. This is defined in the specification and should not need to be written beyond its initialized value.

ENABLE: _YES will allow EESS signaling during the window of opportunity. _NO will prevent EESS signaling.

Offset: 0x46 │ Byte Offset: 0x118 │ Read/Write: R/W │ Reset: 0x82000210 (0b1xxxxx1000000000xxxxxx1000010000)

Bit	Reset	Description
31	YES	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
25:16	0x200	START
9:0	0x210	END

23.6.50 HDMI_NV_PDISP_HDMI_GCP_CTRL_0

The following registers are used to send a single-byte packet intended for sending the general control packet as described in section 5.3.6 of the HDMI specification. This control packet is used to control the AVMUTE flag.

HDMI_GCP_CTRL

This register controls the frequency and generation of GCP packets. The fields of this register are identical to HDMI_AUDIO_INFOFRAME_CTRL.

All four subpackets contain the same GCP packet.

Offset: 0x47 │ Byte Offset: 0x11c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxx0xxx0)

Bit	Reset	Description
8	DIS	SINGLE: 0 = DIS 1 = EN
4	DIS	OTHER: 0 = DIS 1 = EN
0	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

23.6.51 HDMI_NV_PDISP_HDMI_GCP_STATUS_0

HDMI_GCP_STATUS

The SENT bit will be set to _DONE, after the first packet is sent. After the ENABLE bit in INFOFRAME_CTRL is set to _NO, the SENT bit will be set to _WAITING after the start of the next frame to indicate that it is safe to change the contents of the packet header and subpacket registers.

Offset: 0x48 │ Byte Offset: 0x120 │ Read/Write: RO │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	SENT: 1 = DONE 0 = WAITING

23.6.52 HDMI_NV_PDISP_HDMI_GCP_SUBPACK_0

HDMI_GCP_SUBPACK

This register should be written with the contents of the general control packet.

See chapter 5.3.6 of the HDMI specification for more information.

Offset: 0x49 │ Byte Offset: 0x124 │ Read/Write: R/W │ Reset: 0x00000001 (0bxxxxxxxx00000000000000000000001)

Bit	Reset	Description
23:16	0x0	SB2
15:8	0x0	SB1
7:0	0x1	SB0: 1 = SET_AVMUTE 16 = CLR_AVMUTE

23.6.53 HDMI_NV_PDISP_HDMI_CHANNEL_STATUS1_0

HDMI_CHANNEL_STATUS1

This register is used for debug purposes only. Normally, the channel status bits encoded in the SPDIF sample stream are passed directly into the HDMI audio packet. HDMI_CHANNEL_STATUS1/2 can be used to override these bits with user defined values.

If STATUS2_ENABLE is set to _YES, then the 40-bit contents of CHANNEL_STATUS2/1 is inserted into the Cl and Cr bits of the audio sample packets for audio frames 0-39. Cl and Cr contain the channel status bits. See chapter 5.3.4 of the HDMI specification for information on Cr and Cl.

The fields in SPDIF_CHN_STATUS1/2 correspond to the fields in CHANNEL_STATUS1/2 for all values except for the ABCDM field. The values read from SPDIF_CHN_STATUS1/2 can be passed directly into the fields for CHANNEL_STATUS1/2 without any manipulation. This could be used if the user only needs to force one of the fields to a specific value and leave the rest untouched. Information on the channel status bits can be found in Chapter 5 of the IEC60958-3 specification.

ABCDM: This is the first byte of the channel status information. These bits have several meanings. See chapter 5.2.1 of IEC60958-3 for more information.

CODE: This byte defines the Category Code of the input device. This code indicates what type of equipment is generating the input. See 5.3.1 of the IEC60958-3 specification for the values of codes.

SOURCE: Source number of the audio.

CHANNEL: Channel number of the audio. The channel number inserted into channel 2 will be STATUS1_CHANNEL + 1

SFREQ: The reported sampling frequency of the audio stream.

ACCURACY: Transmitter Clock accuracy.

- LVL1 High accuracy mode: Transmitted sampling frequency is within a tolerance of +/-50*10^-6
- LVL2 Normal accuracy mode: Transmitted sampling frequency is within a tolerance of +/-1000*10^-6
- LVL3 Variable pitch shifted clock mode: Only specially designed receivers can receive the signal in this mode.
- OTHER: Interface frame rate not matched to sampling frequency: This is for all other modes that do not have an embedded sampling frequency clock.

Offset: 0x4a │ Byte Offset: 0x128 │ Read/Write: R/W │ Reset: 0xf2ffffff (0b11110010111111111111111111111111)

Bit	Reset	Description
31:28	0xf	ACCURACY: 1 = LVL1 0 = LVL2 2 = LVL3 3 = OTHER
27:24	0x2	SFREQ: 1 = UNDEFINED
23:20	0xf	CHANNEL
19:16	0xf	SOURCE
15:8	0xff	CODE
7:0	0xff	ABCDM

23.6.54 HDMI_NV_PDISP_HDMI_CHANNEL_STATUS2_0

HDMI_CHANNEL_STATUS2

MAX_LENGTH: Reports if the maximum audio sample word length is 20 bits or 24 bits.

LENGTH: Reports the audio sample word length of this block. The value of these bits is dependent on the value of MAX_LENGTH

If MAX_LENGTH = _20 then refer to the MAX20_* defines.

If MAX_LENGTH = _24 then refer to the MAX24_* defines.

All other bit combinations are reserved.

ORIGINAL: Defines the original sampling frequency of the audio stream.

ENABLE: _YES: override the channel status data with the value of these registers. _NO: Send the original channel status data

Offset: 0x4b │ Byte Offset: 0x12c │ Read/Write: R/W │ Reset: 0x000000f1 (0b0xxxxxxxxxxxxxxxxxxxxxxxx11110001)

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
7:4	0xf	ORIGINAL: 0 = UNDEFINED

Bit	Reset	Description
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

23.6.55 HDMI_NV_PDISP_HDMI_EMU0_0

HDMI_EMU0 and 1 do indirect addressing. EMU0 has the address, and EMU1 has the data.

To write:

1. Write the data register with data.

2. Write the address register with bit 31=1, and the address in bits 15:0.

3. Write the address register with bit 31=0.

To read:

1. Write the address register with bit 31=0 and the address in bits 15:0.

2. Read back the address register to introduce delay allowing the read data to propagate.

3. Read the data register.

Offset: 0x4c │ Byte Offset: 0x130 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	regx

23.6.56 HDMI_NV_PDISP_HDMI_EMU1_0

Offset: 0x4d │ Byte Offset: 0x134 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	regx

23.6.57 HDMI_NV_PDISP_HDMI_EMU1_RDATA_0

Offset: 0x4e │ Byte Offset: 0x138 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	0x0	rdata

23.6.58 HDMI_NV_PDISP_HDMI_SPARE_0

HDMI_SPARE

HW_CTS - If this is set to _ENABLE, Hardware measured CTS will be enabled. This method is more accurate than using software controlled CTS. The HDMI block will count the number of TMDS clocks that occur during the interval 1/(128 * audio sample rate) and use the value as CTS and issue the ACR Packet at each interval. When hardware measured CTS is enabled, all ENABLE fields of ACR_CTRL must be set to _NO. Also, ACR_0441_SUBPACK_HIGH_ENABLE must be set to _YES.

ACR_0441_SUBPACK_HIGH_N should be written with the value of N for the current sampling frequency.

The N value must also be written into AUDIO_N_VALUE (See the AUDIO register section for more details).

The value of N can be determined from the current audio sampling frequency, the current pixel clock rate, and tables 7-1. 7-2, and 7-3 in the HDMI specification. In most cases the value in the "Other" row can be used.

FORCE_SW_CTS: When HW_CTS = ENABLE, it uses measured CTS. However, when the audio clock / video clock ratio is known, this is undesirable. Currently, HW_CTS cannot be disabled.

When FORCE_SW_CTS=ENABLE along with HW_CTS=ENABLE, the HW_CTS is used to control transmission of ACR packets, but the CTS is programmed by software in the ACR_0441_SUBPACK_LOW registers.

SUPRESS_SP_B: Default SUPRESS_SP_B=0 is to disable B bits in audio sample packets, for non-present samples. Setting SUPRESS_SP_B=1 restores the old behavior.

CTS_RESET_VAL: When an ACR packet is sent, the CTS counter is reset to the value in this field. This should be set to the INIT value of 1

ACR_PRIORITY: This controls the priority of the ACR packet with respect to Audio Sample packets. This should be set to _HIGH.

LOW: ACR packets will have lower priority than Audio Sample packets

HIGH: ACR packets will have higher priority than Audio Sample packets

Software needs to make sure these registers are set when audio is active:

- HDMI_SPARE_HW_CTS = _HW_CTS_ENABLE
- HDMI_SPARE_CTS_RESET_VAL = 1
- HDMI_SPARE_ACR_PRIORITY = _HIGH
- ACR_CTRL_*_ENABLE = _NO
- ACR_0441_SUBPACK_HIGH_ENABLE = _YES
- ACR_0441_SUBPACK_HIGH_N = N
- AUDIO_N_VALUE = N
- ACR_0441_SUBPACK_LOW_SB1 = _USE_HW_CTS_VAL

HDMI_SPARE: 0:0 rw HW_CTS init=DISABLE

enum (DISABLE,ENABLE)

1:1 rw FORCE_SW_CTS init=DISABLE

enum (DISABLE,ENABLE)

2:2 rw SUPRESS_SP_B init=SUPRESS

enum (SUPRESS, KEEP)

18:16 rw CTS_RESET_VAL init=1

31:31 rw ACR_PRIORITY init=HIGH

enum (HIGH, LOW)

Offset: 0x4f │ Byte Offset: 0x13c │ Read/Write: R/W │ Reset: 0x00010000 (0b00000000000000010000000000000000)

Bit	Reset	Description
31:0	0x10000	Reserved.

23.6.59 HDMI_NV_PDISP_HDMI_SPDIF_CHN_STATUS1_0

HDMI_SPDIF_CHN_STATUS1

HDMI_SPDIF_CHN_STATUS1/2 contains the value of the channel status bits extracted from the incoming S/PDIF audio data stream. See IEC60958-3 chapter 5 for more information on these fields.

USE Specifies consumer use (CONSUMER) or professional (PRO) use of the channel status block. SPDIF_CHN_STATUS1/2 assumes consumer use.

TYPE: Specifies the type of data the audio word represents.

- PCM: Audio sample word represents linear PCM samples
- OTHER: Audio sample word is something other than liner PCM (i.e., compressed audio)

COPYRIGHT: Copyright status of the audio.

- YES: Copyright is asserted
- NO: Copyright is not asserted

D: The values of these bits have different meanings based on the value of _TYPE.

If _TYPE==_PCM, then:

- NO_PREEMPHASIS: 2 audio channels without pre-emphasis
- PREEMPHASIS: 2 audio channels with 50 microseconds/15 microseconds pre-emphasis

All other states are reserved for future use.

MODE Defines one of four possible channel status formats for bytes 1-23 of channel status. Currently only the value "0" is defined. This format is assumed for the SPDIF_CHN_STATUS1/2.

CODE: This byte defines the Category Code of the input device. This code indicates what type of equipment is generating the input. See 5.3.1 of the IEC60958-3 specification for the values of codes.

SOURCE: Source number of the audio. If this value is zero then no source number was reported.

CHANNEL: Channel number of the audio. This reports the channel number reported for the first subframe of audio. If this value is zero then the channel number was not reported.

SFREQ: This is the reported sampling frequency of the input audio stream. The value UNDEFINED means that the sampling frequency was not indicated by the audio stream.

ACCURACY: Transmitter Clock accuracy.

- LVL1 High accuracy mode: Transmitted sampling frequency is within a tolerance of +/-50*10^-6
- LVL2 Normal accuracy mode: Transmitted sampling frequency is within a tolerance of +/-1000*10^-6
- LVL3 Variable pitch shifted clock mode: Only specially designed receivers can receive the signal in this mode

- OTHER: Interface frame rate not matched to sampling frequency: This is for all other modes that do not have an embedded sampling frequency clock.

Offset: 0x50 | Byte Offset: 0x140 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	ACCURACY: 1 = LVL1 0 = LVL2 2 = LVL3 3 = OTHER
27:24	X	SFREQ: 1 = UNDEFINED
23:20	X	CHANNEL: 0 = UNDEFINED
19:16	X	SOURCE: 0 = UNDEFINED
15:8	X	CODE
7:6	X	MODE
5:3	X	D: 0 = NO_PREEMPHASIS 1 = PREEMPHASIS
2	X	COPYRIGHT: 0 = YES 1 = NO
1	X	TYPE: 0 = PCM 1 = OTHER
0	X	USE: 0 = CONSUMER 1 = PRO

23.6.60 HDMI_NV_PDISP_HDMI_SPDIF_CHN_STATUS2_0

HDMI_SPDIF_CHN_STATUS2

MAX_LENGTH: Reports if the maximum audio sample word length is 20 bits or 24 bits.

LENGTH: Reports the audio sample word length of this block. The value of these bits is dependent on the value of MAX_LENGTH

- If MAX_LENGTH=_20 then refer to the MAX20_* defines.
- If MAX_LENGTH=_24 then refer to the MAX24_* defines.

All other bit combinations are reserved.

ORIGINAL: The original sampling frequency of the audio data. UNDEFINED indicates that the original sampling frequency was not defined

Offset: 0x51 | Byte Offset: 0x144 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:4	X	ORIGINAL: 0 = UNDEFINED

Bit	Reset	Description
3:1	X	LENGTH: 0 = MAX20_UNDEF 1 = MAX20_16BITS 2 = MAX20_18BITS 4 = MAX20_19BITS 5 = MAX20_20BITS 6 = MAX20_17BITS 0 = MAX24_UNDEF 1 = MAX24_20BITS 2 = MAX24_22BITS 4 = MAX24_23BITS 5 = MAX24_24BITS 6 = MAX24_21BITS
0	X	MAX_LENGTH

23.6.61 HDMI_NV_PDISP_CRC_CONTROL_0

CRC_CONTROL

The main CRC enable bit flows along the pipeline. This register controls the injection of this bit at the top of the pipeline. This register is double-buffered. The active value is updated at start of each frame from this 'ARM' register.

Other registers such as SOR_STATE2.ASY_CRCMODE and SOR_*CRC* control the actual CRC logic, once enabled.

ARM_CRC_CONTROL enables or disables computation of CRC, effective at the start of next frame.

Offset: 0x96 │ Byte Offset: 0x258 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	NO	ARM_CRC_ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

23.6.62 HDMI_NV_PDISP_INPUT_CONTROL_0

INPUT_CONTROL

HDMI_SRC_SELECT selects from which of the two display units to take input. This register should be changed only when HDMI is idle.

ARM_RGB_RANGE controls whether R/G/B values of 0 and 255 are permitted (FULL), or removed by clamping to [1,254] (LIMITED).

According to the EIA/CEA-861-B and HDMI specifications, the 640x480 VGA mode uses FULL, and all others use LIMITED.

Note that this does not scale the video or change the black and white points (VGA is [0,255], others are [16,235]). That must be done, if necessary, in display or at the source.

See the HDMI 1.2a specification, section 6.6.

This register is double-buffered and will take effect on next frame boundary.

Offset: 0x97 │ Byte Offset: 0x25c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00) | Default: 0x00000001

Bit	Reset	Default	Description
1	FULL	NONE	ARM_VIDEO_RANGE: 0 = FULL 1 = LIMITED
0	DISPLAY	DISPLAYB	HDMI_SRC_SELECT: 0 = DISPLAY 1 = DISPLAYB

23.6.63 HDMI_NV_PDISP_SCRATCH_0

SCRATCH

This register is not used by hardware. It is available for software to store the state or for testing purposes.

Offset: 0x98 │ Byte Offset: 0x260 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA

23.6.64 HDMI_NV_PDISP_PE_CURRENT_0

NV_PDISP_PE_CURRENT

- PE_CURRENT3_3.0
- PE_CURRENT2_3.0
- PE_CURRENT1_3.0
- PE_CURRENT0_3.0

Individual lane Pre-emphasis Current Control (4 bits per lane). The mapping of register value to current is as follows:

- 0000: 0.0mA
- 0001: 1.0mA
- 0010: 2.0mA
- 0011: 3.0mA
- 0100: 4.0mA
- 0101: 5.0mA
- 0110: 6.0mA
- 0111: 7.0mA
- 1000: 8.0mA
- 1001: 9.0mA
- 1010: 10mA
- 1011: 11mA
- 1100: 12mA
- 1101: 13mA
- 1110: 14mA
- 1111: 15mA

See related pre-emphasis control registers above in the NV_PDISP_SOR_PLL0 register.

Offset: 0x99 │ Byte Offset: 0x264 │ Read/Write: R/W │ Reset: 0x08080808 (0bxxxx1000xxxx1000xxxx1000xxxx1000) | Default: 0x00000000

Bit	Reset	SW Default	Description
27:24	0x8	0x0	PE_CURRENT3
19:16	0x8	0x0	PE_CURRENT2
11:8	0x8	0x0	PE_CURRENT1
3:0	0x8	0x0	PE_CURRENT0

23.6.65 HDMI_NV_PDISP_KEY_CTRL_0

HDCP KEY SRAM Register Control

This is the register space for the HDCP ROM interface control register. This register controls writing the contents of the hdcp_key_data bus (56 bits) into the local key store. The keys are decoded in the Crypto block, and then the key data is presented on the bus.

LOCAL: If this bit is ENABLED the on-chip HDCP key store will be used by the HDCP encryption block. If DISABLED, external Crypto-ROM will be used.

AUTOINC: If this bit is ENABLED, the address written to by the WRITE16 function will auto-increment after the operation is complete. This is the normal operating mode.

WRITE16: If this bit is set to TRIGGER, the HDCP keys module will write all 16 bytes of data from the hdcp_key_data bus (sourced by the CD module) into the local key store. The bytes are written into the store at contiguous bytes pointed to by the ADDRESS field in auto-increment mode, or contiguous bytes pointed to by the LOAD_ADDRESS field otherwise. Poll this bit until it reports DONE to ensure the write is complete.

PKEY_REQUEST_RELOAD: Requests that the private key be requested again from KFUSE. Will autoclear to zero as soon as the requested transfer of the key begins. ,Only PKEY_LOADED will indicate when the key is ready for use.

PKEY_LOADED: Indicates that the private key value has been received from KFUSE and is ready for use.

LOAD_ADDRESS: For the WRITE16 function, this field selects the start byte address of the contiguous locations in the local key store to be written with the hdcp_key_data. This field is ignored if the AUTOINC bit is ENABLED. Addresses start at 0.

ADDRESS: This read-only field reports the next byte address in the local key store to be written to once the TRIGGER bits are DONE. For the WRITE16 operation, the address will increment by 16. Addresses start at 0.

Typical operation is as follows:

1. Write to the register with LOCAL_ENABLED, AUTOINC_DISABLED, WRITE5_INIT, WRITE7_INIT, LOAD_ADDRESS_ZERO.
2. Write the downstream KSV key data into the CD module and decrypt the key (see dev_cd.ref - HDCP_KEY_REG(i) and NV_PCIPHER_CTL1).
3. Write to the register with LOCAL_ENABLED, AUTOINC_ENABLED, WRITE5_TRIGGER, WRITE7_INIT, LOAD_ADDRESS_ZERO. Poll for WRITE5_DONE.
4. Write the first downstream key data into the CD module and decrypt the key (see dev_cd.ref - HDCP_KEY_REG(i) and NV_PCIPHER_CTL1).
5. Write to the register with LOCAL_ENABLED, AUTOINC_ENABLED, WRITE7_TRIGGER, WRITE7_INIT, LOAD_ADDRESS_ZERO. Poll for WRITE7_DONE.
6. Repeat steps 4 and 5 39 more times. There are always 40 keys and one KSV value.

If upstream key authentication is required do the following:

7. Write the upstream KSV key data into the CD module and decrypt the key.

8. Write to the register with LOCAL_ENABLED, AUTOINC_ENABLED, WRITE5_TRIGGER, WRITE7_INIT, LOAD_ADDRESS_ZERO. Poll for WRITE5_DONE.
9. Write the first upstream key data into the CD module and decrypt the key (see dev_cd.ref - HDCP_KEY_REG(i) and NV_PCIPHER_CTL1).
10. Write to the register with LOCAL_ENABLED, AUTOINC_ENABLED, WRITE7_TRIGGER, WRITE7_INIT, LOAD_ADDRESS_ZERO. Poll for WRITE7_DONE.
11. Repeat steps 9 and 10 39 more times. There are always 40 keys and one KSV value.

This will leave the store as follows (the required format):

- Byte 0 to Byte 4: downstream KSV
- Byte 5 to Byte 11: 1st downstream key
- Byte 12 to Byte 18: 2nd downstream key

  ...
- Byte 278 to Byte 284: 40th downstream key
- Byte 285 to Byte 289: upstream KSV
- Byte 290 to Byte 296: 1st upstream key

  ...
- Byte 563 to Byte 569: 40'th upstream key

Offset: 0x9a │ Byte Offset: 0x268 │ Read/Write: R/W │ Reset: 0xXXX000X0 (0bxxxxxxxxxx0000000000xxxxxx00xx00)

Bit	R/W	Reset	Description
31:22	RO	X	ADDRESS
21:12	RW	0x0	LOAD_ADDRESS
6	RO	X	PKEY_LOADED: 0 = FALSE 1 = TRUE
5	RW	IDLE	PKEY_REQUEST_RELOAD: 0 = IDLE 1 = TRIGGER
4	RW	DONE	WRITE16: 0 = DONE 1 = TRIGGER 1 = PENDING
1	RW	DISABLED	AUTOINC: 0 = DISABLED 1 = ENABLED
0	RW	DISABLED	LOCAL_KEYS: 0 = DISABLED 1 = ENABLED

23.6.66 HDMI_NV_PDISP_KEY_DEBUG0_0

Offset: 0x9b │ Byte Offset: 0x26c │ Read/Write: R/W │ Reset: 0x000000X0 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0xxx0)

Bit	R/W	Reset	Description
6	RO	X	CHECKSUMCMP_HIGH: 0 = MISMATCH 1 = MATCH
5	RO	X	CHECKSUMCMP_LOW:

Bit	R/W	Reset	Description
			0 = MISMATCH 1 = MATCH
4	RW	DONE	CHECKSUM: 0 = DONE 1 = TRIGGER 1 = PENDING
0	RW	DONE	SRAMCLEAR: 0 = DONE 1 = TRIGGER 1 = PENDING

23.6.67 HDMI_NV_PDISP_KEY_DEBUG1_0

Offset: 0x9c | Byte Offset: 0x270 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	CHECKSUMVAL_HIGH
15:0	0x0	CHECKSUMVAL_LOW

23.6.68 HDMI_NV_PDISP_KEY_DEBUG2_0

Offset: 0x9d | Byte Offset: 0x274 | Read/Write: R/W | Reset: 0xXXXXX000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0xx0x)

Bit	Reset	Description
31:24	X	SRAMDATA
21:12	X	SRAMADDR
4	DONE	SRAMWRITE1: 0 = DONE 1 = TRIGGER 1 = PENDING
1	DISABLED	SRAMAUTOINC: 0 = DISABLED 1 = ENABLED

23.6.69 HDMI_NV_PDISP_KEY_HDCP_KEY_0_0

Offset: 0x9e | Byte Offset: 0x278 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	KEY_REG

23.6.70 HDMI_NV_PDISP_KEY_HDCP_KEY_1_0

Offset: 0x9f | Byte Offset: 0x27c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	KEY_REG

23.6.71 HDMI_NV_PDISP_KEY_HDCP_KEY_2_0

Offset: 0xa0 | Byte Offset: 0x280 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	KEY_REG

23.6.72 HDMI_NV_PDISP_KEY_HDCP_KEY_3_0

Offset: 0xa1 | Byte Offset: 0x284 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	KEY_REG

23.6.73 HDMI_NV_PDISP_KEY_HDCP_KEY_TRIG_0

Offset: 0xa2 | Byte Offset: 0x288 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx)

Bit	Reset	Description
8	IDLE	LOAD_HDCP_KEY: 0 = IDLE 1 = TRIGGER

23.6.74 HDMI_NV_PDISP_KEY_SKEY_INDEX_0

Sixteen (16) sets of AES keys are available. The convention is:

- 15: test/debug
- 0-14: production

Offset: 0xa3 | Byte Offset: 0x28c | Read/Write: WO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	IDX_VALUE: 15 = TEST

23.6.75 HDMI_NV_PDISP_INT_STATUS_0

Sticky interrupt status, write 1 to clear

Interrupt support: Each interrupt event has 3 configuration states:

MASK	ENABLE	Result
x	0	nothing
0	1	INT_STATUS asserted
1	1	INT_STATUS asserted, and interrupt pin asserted

Offset: 0xcc | Byte Offset: 0x330 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3	X	SCRATCH: Software has written NV_PDISP_SCRATCH register (bit 31 changed)
2	X	CP_REQUEST: HDA Codec has written CP_REQUEST register
1	X	CODEC_SCRATCH1: HDA Codec has written CODEC_SCRATCH1 register (bit 31 changed)

Bit	Reset	Description
0	X	CODEC_SCRATCH0: HDA Codec has written CODEC_SCRATCH0 register (bit 31 changed)

23.6.76 HDMI_NV_PDISP_INT_MASK_0

MASKED prevents the interrupt from asserting the HDMI interrupt pin to the CPU, but still allows the interrupt to appear in INT_STATUS. NOTMASKED allows the interrupt to assert, assuming INT_ENABLE is true also.

Offset: 0xcd │ Byte Offset: 0x334 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	MASKED	SCRATCH_MASK: 0 = MASKED 1 = NOTMASKED
2	MASKED	CP_REQUEST_MASK: 0 = MASKED 1 = NOTMASKED
1	MASKED	CODEC_SCRATCH1_MASK: 0 = MASKED 1 = NOTMASKED
0	MASKED	CODEC_SCRATCH0_MASK: 0 = MASKED 1 = NOTMASKED

23.6.77 HDMI_NV_PDISP_INT_ENABLE_0

ENABLE allows the event to appear in INT_STATUS, and allows the interrupt to signal the CPU, assuming INT_MASK=NOTMASKED.

Offset: 0xce │ Byte Offset: 0x338 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	DISABLE	SCRATCH_ENABLE: 0 = DISABLE 1 = ENABLE
2	DISABLE	CP_REQUEST_ENABLE: 0 = DISABLE 1 = ENABLE
1	DISABLE	CODEC_SCRATCH1_ENABLE: 0 = DISABLE 1 = ENABLE
0	DISABLE	CODEC_SCRATCH0_ENABLE: 0 = DISABLE 1 = ENABLE

23.6.78 HDMI_NV_PDISP_HDMI_VSI_INFOFRAME_CTRL_0

HDMI_VSI_CTRL

This register controls the frequency and generation of the Vendor Specific infoframe packets. The Vendor Specific Infoframe packet contains supplemental information about the raster type. The fields of this register are identical to HDMI_AVI_INFOFRAME_CTRL except where noted below.

The Vendor Specific Infoframe is described in Appendix H of the HDMI 1.4a specification.

- 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 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 both OTHER and SINGLE to _EN.

- CHKSUM_HW: Hardware provides a way to calculate the Checksum for the infoframes.
  - _ENABLE will enable the hardware calculation to be passed to the packet
  - _DISABLE will use the register value defined in SF_HDMI_VSI_SUBPACK0_LOW_PB0 for the checksum.
- VIDEO_FMT

  The Vendor Specific Infoframe contains an HDMI_Video_Format Field. This field specifies how the remaining bytes in the infoframe should be interpreted by the specification. When switching to a 3D stereo format, the HDMI_Video_Format field and the 3D_Structure field need to be set appropriately. This priv register field selects whether hardware or software will set those fields.
- _SW_CONTROLLED: The HDMI_Video_Format field and 3D_Structure field will be set by SF_HDMI_VSI_SUBPACK0_HIGH_PB4 and PB5.
- _HW_CONTROLLED: The HDMI_Video_Format field and 3D_Structure field will be set automatically by Hardware based on the state of the NV_917D_Core_Head_SetHdmiCtrl method.

```
31           24 23             15         10  8 7     4       0
.-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-.
|0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0| |0 0 0| |0 0 0| |0 0 0| |0 0 0| | SF_HDMI_VSI_CTRL
`-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-'
```

Offset: 0xd6 | Byte Offset: 0x358 | Read/Write: R/W | Reset: 0x00010200 (0bxxxxxxxxxxxxxxxx1xxxxxx10xxx0xxx0)

Bit	Reset	Description
16	HW_CONTROLLED	VIDEO_FMT: 0 = SW_CONTROLLED 1 = HW_CONTROLLED
9	ENABLE	CHKSUM_HW: 0 = DISABLE 1 = ENABLE 0 = DIS 1 = EN
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

23.6.79 HDMI_NV_PDISP_HDMI_VSI_INFOFRAME_STATUS_0

HDMI_VSI_STATUS

The SENT bit will be set to _DONE, after the first packet is sent. After the ENABLE bit in VSI_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.

```
31          24 23          16 15          8 7           0
.-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-.
|0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0| | SF_HDMI_VSI_STATUS
`-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-'
```

Offset: 0xd7 | Byte Offset: 0x35c | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	SENT: 0 = WAITING 1 = DONE

23.6.80 HDMI_NV_PDISP_HDMI_VSI_INFOFRAME_HEADER_0

HDMI_VSI_HEADER

This register should be written with the contents of the packet header.

```
31          24 23             15         10  8 7     4       0
.-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-.
|0 0 0 0 0 0 0 0|     HB2       |    HB1        |    HB0        | SF_HDMI_VSI_HEADER
`-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-'
```

Offset: 0xd8 | Byte Offset: 0x360 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:16	0x0	HB2
15:8	0x0	HB1
7:0	0x0	HB0

23.6.81 HDMI_NV_PDISP_HDMI_VSI_INFOFRAME_SUBPACK0_LOW_0

HDMI_VSI_SUBPACK0_LOW

Bytes 0-3 of the packet are written into this register.

```
31          24 23             15         10  8 7     4       0
.-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-.
|      PB3      |     PB2       |    PB1        |    PB0        | SF_HDMI_VSI_SUBPACK0_LOW
`-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-'
```

Offset: 0xd9 | Byte Offset: 0x364 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB3
23:16	0x0	PB2
15:8	0x0	PB1
7:0	0x0	PB0

23.6.82 HDMI_NV_PDISP_HDMI_VSI_INFOFRAME_SUBPACK0_HIGH_0

HDMI_VSI_SUBPACK0_HIGH

Bytes 4-6 of the packet are written into this register.

```
31           24 23             15         10  8 7     4       0
.-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-.
|0 0 0 0 0 0 0 0|     PB6       |    PB5        |    PB4        | HDMI_VSI_INFOFRAME_SUBPACK0_HIGH
`-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-'
```

Offset: 0xda | Byte Offset: 0x368 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB6
15:8	0x0	PB5
7:0	0x0	PB4

23.6.83 HDMI_NV_PDISP_HDMI_VSI_INFOFRAME_SUBPACK1_LOW_0

HDMI_VSI_SUBPACK1_LOW

Bytes 7-10 of the packet are written into this register.

```
31           24 23             15         10  8 7     4       0
.-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-.
|      PB10     |     PB9       |    PB8        |    PB7        | HDMI_VSI_INFOFRAME_SUBPACK1_LOW
`-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-'
```

Offset: 0xdb | Byte Offset: 0x36c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB10
23:16	0x0	PB9
15:8	0x0	PB8
7:0	0x0	PB7

23.6.84 HDMI_NV_PDISP_HDMI_VSI_INFOFRAME_SUBPACK1_HIGH_0

HDMI_VSI_SUBPACK1_HIGH

Bytes 11-13 of the packet are written into this register.

```
31           24 23             15         10  8 7     4       0
.-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-.
|0 0 0 0 0 0 0 0|     PB13      |    PB12       |    PB11       | HDMI_VSI_INFOFRAME_SUBPACK1_HIGH
`-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-'
```

Offset: 0xdc | Byte Offset: 0x370 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB13
15:8	0x0	PB12
7:0	0x0	PB11

23.6.85 HDMI_NV_PDISP_HDMI_VSI_INFOFRAME_SUBPACK2_LOW_0

HDMI_VSI_SUBPACK2_LOW

Bytes 14-17 of the packet are written into this register.

```
31           24 23             15         10  8 7     4       0
.-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-.
|      PB17     |     PB16      |    PB15       |    PB14       | HDMI_VSI_INFOFRAME_SUBPACK2_LOW
`-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-'
```

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

Bit	Reset	Description
31:24	0x0	PB17
23:16	0x0	PB16
15:8	0x0	PB15
7:0	0x0	PB14

23.6.86 HDMI_NV_PDISP_HDMI_VSI_INFOFRAME_SUBPACK2_HIGH_0

HDMI_VSI_SUBPACK2_HIGH

Bytes 18-20 of the packet are written into this register.

```
31           24 23             15         10  8 7     4       0
.-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-.
|0 0 0 0 0 0 0 0|     PB20      |    PB19       |    PB18       | HDMI_VSI_INFOFRAME_SUBPACK2_HIGH
`-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-'
```

Offset: 0xde | Byte Offset: 0x378 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000)

Bit	Reset	Description
23:16	0x0	PB20
15:8	0x0	PB19
7:0	0x0	PB18

23.6.87 HDMI_NV_PDISP_HDMI_VSI_INFOFRAME_SUBPACK3_LOW_0

HDMI_VSI_SUBPACK3_LOW

Bytes 21-24 of the packet are written into this register.

```
31           24 23             15         10  8 7     4       0
.-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-.
|      PB24     |     PB23      |    PB22       |    PB21       | HDMI_VSI_INFOFRAME_SUBPACK3_LOW
`-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-'
```

Offset: 0xdf | Byte Offset: 0x37c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB24
23:16	0x0	PB23
15:8	0x0	PB22
7:0	0x0	PB21

23.6.88 HDMI_NV_PDISP_HDMI_VSI_INFOFRAME_SUBPACK3_HIGH_0

HDMI_VSI_SUBPACK3_HIGH

Bytes 25-27 of the packet are written into this register.

```
31           24 23             15         10  8 7     4       0
.-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-.
|0 0 0 0 0 0 0 0|     PB27      |    PB26       |    PB25       | HDMI_VSI_INFOFRAME_SUBPACK3_HIGH
`-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-'
```

Offset: 0xe0 | Byte Offset: 0x380 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB27
15:8	0x0	PB26
7:0	0x0	PB25

23.7 Serial Output Resource Registers

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

23.7.1 HDMI_NV_PDISP_SOR_PWR_0

This register contains bits that control the power state of the SOR. For simplicity, the sequencer will be used to perform all power control operations in the SOR (even in TMDS where the sequencing is relatively simple). This unifies the approach and makes it easier for the hardware and software to deal with the SOR's. At boot, the software must load and configure the SOR sequencer via the SOR_SEQ_CTL and SOR_SEQ_INST registers below. The sequencer must be properly programmed for power control to work.

NORMAL_STATE: Sets the normal operating state. There are two choices: powered up and powered down. To change this register, software should load in the new value and write SETTING_NEW=TRIGGER. Once the change has been successfully completed, SETTING_NEW will indicate DONE. This is the state that software will want to control in order to power up and down the interface.

NORMAL_START: The execution of the power up and power down sequence can be modified somewhat by choosing to use either the NORM or ALT program entry point.

SAFE_STATE: Sets the safe operating state. There are only two choices: powered up and powered down. To change this register, software should load in the new value and write SETTING_NEW=TRIGGER. Once the change has been successfully completed, the SETTING_NEW will indicate DONE. The execution of the power up and power down sequence can be modified somewhat by choosing to use either the standard or the alternate program entry point. This is the state that the hardware will use whenever it initiates the mode switch/shutdown procedure for HDMI. In general, this should be set to STATE_PD and left there always.

SAFE_START: The execution of the power up and power down sequence can be modified somewhat by choosing to use either the NORM or ALT program entry point.

HALT_DELAY: Once the sequencer gets to the instruction with the HALT=1, the program is essentially complete; that is, the attached unit is powered up or down as was requested. Some panels, however, have a minimum time before their state can be changed again. If the halt instruction has a non-zero delay, this bit will be set while that time expires.

MODE: The currently active state, normal or safe. After reset, the MODE is always SAFE.

SETTING_NEW: This bit is used to trigger a new setting of power mode to take effect. The typical procedure might be something like:

1. Make sure SETTING_NEW==DONE, i.e., not already an outstanding change request. If there is an outstanding change request, software must wait for it to complete.

2. Update the NORMAL and SAFE power mode fields.

3. Write SETTING_NEW=TRIGGER.

4. Poll for SETTING_NEW=DONE. If it does not happen within one frame time, then software may opt to accelerate the change by writing SOR_SEQ_CTL SWITCH=FORCE (be careful not to disturb other settings in that register!).

Usage: boot / initialization / mode switch / normal operation / shutdown

Offset: 0x55 | Byte Offset: 0x154 | Read/Write: R/W | Reset: 0xXX000000 (0b0xxxxxxxxxxxxx00xxxxxxxxxxxxxx00)

Bit	R/W	Reset	Description
31	RW	0x0	SETTING_NEW: 0 = DONE 1 = PENDING 1 = TRIGGER
28	RO	X	MODE: 0 = NORMAL 1 = SAFE
24	RO	X	HALT_DELAY: 0 = DONE 1 = ACTIVE
17	RW	0x0	SAFE_START: 0 = NORMAL 1 = ALT
16	RW	0x0	SAFE_STATE: 0 = PD 1 = PU
1	RW	0x0	NORMAL_START: 0 = NORMAL 1 = ALT
0	RW	0x0	NORMAL_STATE: The normal state depends on the power sequencing state. NORMAL_STATE=1 is correct for powered-up HDMI. 0 is for the powered-down / reset state. 0 = PD 1 = PU

23.7.2 HDMI_NV_PDISP_SOR_TEST_0

This register contains control bits that configure certain aspects of testing mode of the SOR.

TEST_ENABLE: To enable testing:

- 0 = normal operation
- 1 = test mode

In test mode test_fastclkint and test_loadpulse signals from the core are used in place of the internally generated signals.

INVD: Invert the data. Note that combining INVD with the choice of DSRC and TPAT below permits ramp up, ramp down, walking ones, walking zeros, all ones, all zeros.

ACT_HEAD_OPMODE: Report the current OR operating mode.

ATTACHED: Report whether the OR is currently attached to a head.

DSRC: When DSRC=NORMAL, the serializers behave normally and use the encoded RGB data. When DSRC=DEBUG, the serializers load from DEBUGA0, DEBUGA1, DEBUGB0, and DEBUGB1 bits instead of the normal data coming down the pipe. When DSRC=TGEN, the data is taken from the built in test generator.

TPAT: Selects the test generator pattern. When test generator is running, H will be high for 1 in 4 repetitions (of duration 1024 clocks), V will be high for 4 in 16 repetitions coincident with H. Only operates when test mode is enabled.

LO: force all 0s

TDAT: use hardware fixed value

RAMP: test generator output ramps

WALK: test generator output is a walking one

MAXSTEP: 0, 1023, ...

MINSTEP: 511, 512, ...

CRC: The source of the data for CRC computation (for verification) can be either before the high-speed serializer logic, or after the high-speed serializer/deserializer logic.

TESTMUX[7:0]: Test MUX select - output seen on PROBE

Usage: Debug

Offset: 0x56 │ Byte Offset: 0x158 │ Read/Write: R/W │ Reset: 0x00800X00 (0b000000001000xx00xxxxxxxxx0xxxx0x)

Bit	R/W	Reset	Description
31:24	RW	0x0	TESTMUX: 0 = AVSS 2 = CLOCKIN 4 = PLL_VOL 8 = SLOWCLKINT 16 = AVDD 32 = VDDREG 64 = REGREF_VDDREG 128 = REGREF_AVDD
23	RW	0x1	CRC: 0 = PRE_SERIALIZE 1 = POST_DESERIALIZE
22:20	RW	0x0	TPAT: 0 = LO 1 = TDAT 2 = RAMP 3 = WALK 4 = MAXSTEP 5 = MINSTEP
17:16	RW	0x0	DSRC: 0 = NORMAL 1 = DEBUG 2 = TGEN
10	RO	X	ATTACHED: 0 = FALSE 1 = TRUE
9:8	RO	X	ACT_HEAD_OPMODE: 0 = SLEEP 1 = SNOOZE 2 = AWAKE
6	RW	0x0	INVD: 0 = DISABLE 1 = ENABLE
1	RW	0x0	TEST_ENABLE: 0 = DISABLE 1 = ENABLE

23.7.3 HDMI_NV_PDISP_SOR_PLL0_0

These registers configure the main SOR PLL and other frequency-dependent controls. The value loaded is a function of the pixel clock frequency, and whenever the pixel clock changes, these registers must be updated. In general, these registers will be updated during the second interrupt of a mode switch.

PWR: 1 = power down the TMDS PLL

PDBG: 1 = power down the bandgap

VCOPD: 1 = power down the VCO

PDPORT: 1 = power down the output drivers

RESISTORSEL: Selects the internal resistor (0) or external resistor (1).

PULLDOWN: Weak pull-down enable

- 1 = 2Kohm pulldown on all outputs.
- 0 = Weak pulldown disabled.

Note: pulldown is also controlled by the sequencer. the pulldown is derived from an OR of:

NV_PDISP_SOR_PLL0_PULLDOWN and the sequencer control of pulldown. This priv register field does not read the final pull-down value. In other words if software writes NV_PDISP_SOR_PLL0_PULLDOWN to DISABLED and sequencer pulldown is enabled, then an NV_PDISP_SOR_PLL0_PULLDOWN read will return DISABLED (while pulldown might be ENABLED due to sequencer control).

VCOCAP[3:0]: Selects the VCO capacitor and adjusts ring oscillator inter-stage load.

FILTER[3:0]: Selects the loop filter and adjusts the filter resistor value.

ICHPMP[3:0]: Specifies additions to the charge pump current in steps of 0.375 µA.

TMDS_TERM: This bit is used to enable termination. Termination is only used in TMDS mode of operation, and may not be needed at lower operating frequencies (in which case, disabling it saves power).

TERMADJ[3:0]: Termination resistance control.

LOADADJ[3:0]: Load pulse position adjust.

AUX0-AUX7: Most of these bits currently have no assigned function, but are provided to permit control of possible future features of the macro.

- AUX0: No function
- AUX1: No function
- AUX2: No function
- AUX3: Rotate green channel by 1 bit to reduce TMDS EMI
- AUX4: No function
- AUX5: No function
- AUX6: No function
- AUX7: No function

TMDS_ANALOG_X4_HP_B revision:

BG_V17_S[3:0]: Bandgap 1.7V output voltage control

TX_REG_LOAD[1:0]: TX regulator default loading

- 00: 0.5mA (default)
- 01: 1.0mA
- 10: 1.5mA
- 11: 2.0mA

PE_EN: Pre-emphasis enable. PE_EN active only at 1080p, but turn off at 720p

- 0: Disable
- 1: Enable

HALF_FULL_PE: Pre-emphasis half bit time or full bit time (Note 2)

- 0: Half bit time (default)
- 1: Full bit time (for long trace)

S_D_PIN_PE: Pre-emphasis on one single pin or on differential pins (D+ and D-) (Note 2)

- 0: Single pin (default)
- 1: Differential pins

See also the NV_PDISP_PE_CURRENT register below.

Usage: boot / initialization / mode switch

Offset: 0x57 │ Byte Offset: 0x15c │ Read/Write: R/W │ Reset: 0x0200333f (0bxx000010xxxx000000110011xx111111)| Default: 0x01000000

Bit	Reset	Default	Description
29:28	0x0	NONE	TX_REG_LOAD: 0 = RESETV
27:24	0x2	0x1	ICHPMP: 2 = RESETV
19:16	0x0	NONE	FILTER: 0 = RESETV
15:12	0x3	NONE	BG_V17_S: 3 = RESETV
11:8	0x3	NONE	VCOCAP: 3 = RESETV
5	0x1	DISABLE	PULLDOWN: 0 = DISABLE 1 = ENABLE
4	0x1	NONE	RESISTORSEL: 0 = INT 1 = EXT 1 = RESETV
3	0x1	NONE	PDPORT: 0 = ON 1 = OFF
2	0x1	NONE	VCOPD: 0 = RESCIND 1 = ASSERT
1	0x1		PDBG: 0 = ON 1 = OFF
0	0x1		PWR: 0 = ON 1 = OFF

23.7.4 HDMI_NV_PDISP_SOR_PLL1_0

Usage: boot / initialization / mode switch

Offset: 0x58 │ Byte Offset: 0x160 │ Read/Write: R/W │ Reset: 0x00301200 (0bx000xxxx0011xxxxxxx10010xxxxxxxx)

Bit	Reset	Description
30	SINGLE	S_D_PIN_PE: 0 = SINGLE

Bit	Reset	Description
		1 = DIFFERENTIAL
29	HALF	HALF_FULL_PE: 0 = HALF 1 = FULL
28	DISABLE	PE_EN: 0 = DISABLE 1 = ENABLE
23:20	0x3	LOADADJ: 0 = CENTER
12:9	0x9	TMDS_TERMADJ
8	0x0	TMDS_TERM: 0 = DISABLE 1 = ENABLE

23.7.5 HDMI_NV_PDISP_SOR_PLL2_0

Spare Registers for TMDS Control

AUX3: lane 1 rotation control

- 0: No rotation (Default)
- 1: Rotate right 1 bit

Usage: boot / initialization / mode switch

Offset: 0x59 │ Byte Offset: 0x164 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000xxxxxxxxxxxxxxxx)

Bit	Reset	Description
23	0x0	AUX7
22	0x0	AUX6
21	0x0	AUX5
20	0x0	AUX4
19	0x0	AUX3
18	0x0	AUX2
17	0x0	AUX1
16	0x0	AUX0

23.7.6 HDMI_NV_PDISP_SOR_CSTM_0

The class permits the driver to select a number of operating modes for the SOR. Some of these modes (TMDS modes) are well defined and stable. Some modes may require customization by software before they will work. These registers are used for that customization. The fields available for customization are:

- PD_TXDA[3:0]: Bitwise control to power down the data pins of link A. Set to DISABLE to power down the pin.
- PD_TXDB[3:0]: Bitwise control to power down the data pins of link B. Set to DISABLE to power down the pin.
- PD_TXCA: Power down the clock pin of link A. Set to DISABLE to power down the pin.
- PD_TXCB: Power down the clock pin of link B. Set to DISABLE to power down the pin.

UPPER: Designates whether LVDS bank A is the upper, odd, or first pixel. Bank B is always set to !UPPER. The serial output from UPPER=1 will be Clock and A0-A3. The serial output from UPPER=0 (and DUALMODE) will be Clock and A4-A7. The default is bank A has UPPER=1, bank B has UPPER=0.

MODE[1:0]: Controls the digital output encoding applied to the data stream in custom mode and is only used for data muxing at the input of the SOR. This field does not control the TMDS macro.

- 0 = LVDS
- 1 = TMDS
- 2 = Reserved
- 3 = Reserved

LINKACTA, LINKACTB: Enables (1) or disables (0) the digital logic of links A and B

LVDS_EN: Output driver configuration for controlling the encoding of the data and the output common mode control. This does not control the internal clock dividers of the TMDS macro.

- 0 = TMDS
- 1 = LVDS

DUP_SYNC: This field only has an effect when in LVDS mode. When asserted in LVDS mode, it forces the link to use DE, HSYNC, and VSYNC for the encoding and to never use RES, CNTLE, and CNTLF. RES becomes DE. CNTLE becomes HSYNC. CNTLF becomes VSYNC.

NEW_MODE: For backwards compatibility, set register to 0 so the old mode is used. In old mode, for the second link in dual link mode, all the control bits are zeroed out except for VSYNC. When a new mode is used, none of the control bits of the second link for dual-link mode are zeroed out.

BALANCED: For MODE = LVDS, this enables balanced encoding. Balanced mode will selectively invert sets of bits in the serial stream in an attempt to keep the average DC value near zero. Default is unbalanced. Has no effect in other modes.

PLLDIV: Controls the internal clock dividers of the TMDS_MACRO by setting the feedback divider for the high-speed PLL.

- 0 = divide by 7 (LVDS)
- 1 = divide by 10 (TMDS)

ROTCLK[3:0]: Skews the TXC clock to come out earlier. By changing this register value, you can configure the skew between output data (TX data) and output clock (TXC). This field specifies the number of sclk cycles which the output clock should come out earlier than its normal phase. For TMDS, sclk is a 10x pixel clock, so ROTVAL should be between 0-9. For LVDS, sclk is a 7x pixel clock, so ROTVAL should be between 0-7.

ROTDAT[2:0]: Before encoding the 8 bits of each color channel, the 8 bits within each color channel can be right rotated. All color channels are rotated by the same amount. For example, if ROTDAT = 6, then input channel data (r7,r6,r5,r4,r3,r2,r1,r0} would become {r5,r4,r3,r2,r1,r1,r7,r6}.

ROTDAT should be between 0 and 7.

TMDS modes of operation are standard and stable. The table below summarizes the fixed values the hardware uses for the above fields for TMDS operating modes.

	SINGLE TMDS_A	SINGLE TMDS_B	SINGLE TMDS_AB	SINGLE TMDS	Dual TMDS
PD_TXDA[2:0]	0	7	0	0	0
PD_TXDA[3]	1	1	1	1	1
PD_TXDB[2:0]	7	0	0	0	0
PD_TXDB[3]	1	1	1	1	1
PD_TXCA	0	1	0	0	0

	SINGLE TMDS_A	SINGLE TMDS_B	SINGLE TMDS_AB	SINGLE TMDS	Dual TMDS
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 debug register set. Software should really never need to use it. Also, note that if LVDS_ONLY=TRUE, then this register cannot be used as the hardware will only permit Protocol=LVDS to be enabled. This register is intended to have the same format as the second register.

Usage: boot / initialization / mode switch

Offset: 0x5a │ Byte Offset: 0x168 │ Read/Write: R/W │ Reset: 0x0001c800 (0bx0000000xx0x000111001x0000000000) | Default: 0x02000000

Bit	Reset	Default	Description
30:28	0x0	NONE	ROTDAT: 0 = RESETV
27:24	0x0	0x2	ROTCLK: 0 = RESETV
21	0x0	NONE	PLLDIV: 0 = BY_7 1 = BY_10
19	0x0	NONE	BALANCED: 0 = DISABLE 1 = ENABLE
18	0x0	NONE	NEW_MODE: 0 = DISABLE 1 = ENABLE
17	0x0	NONE	DUP_SYNC: 0 = DISABLE 1 = ENABLE
16	0x1	NONE	LVDS_EN: 0 = DISABLE 1 = ENABLE
15	0x1	NONE	LINKACTB: 0 = DISABLE 1 = ENABLE
14	0x1	NONE	LINKACTA: 0 = DISABLE 1 = ENABLE
13:12	0x0	NONE	MODE: 0 = LVDS 1 = TMDS

Bit	Reset	Default	Description
11	0x1	NONE	UPPER: 1 = TRUE 0 = FALSE
9	0x0	NONE	PD_TXCB: 1 = DISABLE 0 = ENABLE
8	0x0	NONE	PD_TXCA: 1 = DISABLE 0 = ENABLE
7	0x0	NONE	PD_TXDB_3: 1 = DISABLE 0 = ENABLE
6	0x0	NONE	PD_TXDB_2: 1 = DISABLE 0 = ENABLE
5	0x0	NONE	PD_TXDB_1: 1 = DISABLE 0 = ENABLE
4	0x0	NONE	PD_TXDB_0: 1 = DISABLE 0 = ENABLE
3	0x0	NONE	PD_TXDA_3: 1 = DISABLE 0 = ENABLE
2	0x0	NONE	PD_TXDA_2: 1 = DISABLE 0 = ENABLE
1	0x0	NONE	PD_TXDA_1: 1 = DISABLE 0 = ENABLE
0	0x0	NONE	PD_TXDA_0: 1 = DISABLE 0 = ENABLE

23.7.7 HDMI_NV_PDISP_SOR_LVDS_0

The second register defines the LVDS custom mode. This is the base from which all LVDS variants can be customized. This register is intended to have the same format as the first register.

Usage: boot / initialization / mode switch

Offset: 0x5b │ Byte Offset: 0x16c │ Read/Write: R/W │ Reset: 0x00XXXX0X (0bx0000000xxxx000x1xxxxx0x00000xxx)

Bit	R/W	Reset	Description
30:28	RW	0x0	ROTDAT: 0 = RESETV
27:24	RW	0x0	ROTCLK: 0 = RESETV
21	RO	X	PLLDIV: 0 = BY_7
19	RW	0x0	BALANCED: 0 = DISABLE 1 = ENABLE
18	RW	0x0	NEW_MODE:

Bit	R/W	Reset	Description
			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
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

23.7.8 HDMI_NV_PDISP_SOR_CRCA_0

The following three registers are used to fetch CRC's when running VGA mode tests. One contains the valid bit, one contains the actual computed CRC, and the third contains the error (overrun bit). Proper use of these registers is as follows:

1) Poll SOR_CRCA for VALID == TRUE.

2) Reset SOR_CRCA by writing RESET to it.

3) Read SOR_CRCB to get the CRC

Usage: verification

Offset: 0x5c │ Byte Offset: 0x170 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	VALID: 0 = FALSE 1 = TRUE 1 = RESETV

23.7.9 HDMI_NV_PDISP_SOR_CRCB_0

Usage: verification

Offset: 0x5d │ Byte Offset: 0x174 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CRC

23.7.10 HDMI_NV_PDISP_SOR_BLANK_0

This register can be used to override the SOR output resource pixels with blank data.

OVERRIDE: Setting this field to true will override the pixel bus from the RG and output black pixels instead.

TRANSITION: This field controls the timing of the output resource blank override. The choices are IMMEDIATE. The output resource will be blanked or restored to its previous output data immediately.

NEXT_VSYNC The output resource will be blanked or restored to its previous output data at the next vsync.

STATUS This read-only field returns BLANKED when the output resource is sending blank pixels forced by the OVERRIDE bit, otherwise it returns NOT_BLANKED.

Usage: boot / initialization / mode switch / normal operation

Offset: 0x5e │ Byte Offset: 0x178 │ Read/Write: R/W │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	R/W	Reset	Description
2	RO	X	STATUS: 0 = NOT_BLANKED 1 = BLANKED
1	RW	0x0	TRANSITION: 0 = IMMEDIATE 1 = NEXT_VSYNC
0	RW	0x0	OVERRIDE: 0 = FALSE 1 = TRUE

23.7.11 HDMI_NV_PDISP_SOR_SEQ_CTL_0

Sequencer control registers for SOR. Up to three pins can be assigned to an SOR for use in controlling the power to an attached LVDS flat panel. The meaning of any particular pin and whether it is actually available to this SOR is controlled in either the host or PCB manager. In addition, the sequencer can override the individual link clock and data power control pins, thereby forcing them all into disabled or tristate mode and it can override the DE signal from the RG (for TMDS mode) to force the link to become inactive.

PU_PC: The program counter for the start of the power up program sequence. Always 0.

PU_PC_ALT: The alternate entry point into the power up program sequence. Defaults to the same value as PU_PC.

PD_PC: The program counter for the start of the power down program sequence. Defaults to 8, evenly dividing the available program space.

PD_PC_ALT: The alternate entry point into the power down program sequence. Defaults to the same default value as PD_PC.

PC: The current value of the program counter (useful for status and debug).

STATUS: Indicates if the sequencer is STOPPED or RUNNING.

SWITCH: If a particular sequencer instruction is waiting for vsync to arrive, writing this bit to FORCE will cause the wait condition to be satisfied immediately.

> **Note:** The sequencer can begin with arbitrary phase relative to the 1 µs timer that is used to advance the internal counters. Thus the actual time delay for the first event can be almost one microsecond less than what is requested. Subsequent instructions in a chain of events do transition on 1 µs boundaries, however. If a minimum specification of "x" µs is required, then it is best to program "x+1" µs of delay.

Usage: boot / initialization / mode switch

Offset: 0x5f │ Byte Offset: 0x17c │ Read/Write: R/W │ Reset: 0xX00X880X (0bx0xxxxxxxxxxxxxx100010000000xxxx)

Bit	R/W	Reset	Description
30	RW	WAIT	SWITCH: 0 = WAIT 1 = FORCE
28	RO	X	STATUS: 0 = STOPPED 1 = RUNNING
19:16	RO	X	PC
15:12	RW	0x8	PD_PC_ALT
11:8	RW	0x8	PD_PC
7:4	RW	0x0	PU_PC_ALT
3:0	RO	X	PU_PC

23.7.12 HDMI_NV_PDISP_SOR_SEQ_INST0_0

Usage: boot / initialization

Offset: 0x60 │ Byte Offset: 0x180 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000) | Default: 0x00802001

Bit	Reset	Default	Description
31	0x0	NONE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	NONE	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	NONE	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	NONE	BLANK_V: 0 = NORMAL 1 = INACTIVE

Bit	Reset	Default	Description
27	0x0	NONE	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	NONE	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	NONE	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	NONE	TRISTATE_IOS: 0 is the correct value for power-up. 1 is the value when HDMI is off. 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	TRUE	DRIVE_PWM_OUT_LO: Power sequencer; verified with value 1 (TRUE). 0 = FALSE 1 = TRUE
22	0x0	NONE	PIN_B: 0 = LOW 1 = HIGH
21	0x0	NONE	PIN_A: 0 = LOW 1 = HIGH
15	0x1	NONE	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	VSYNC	WAIT_UNITS: Power sequencer; verified with WAIT_UNITS=2 (VSYNC). 0 = US 1 = MS 2 = VSYNC
9:0	0x0	0x1	WAIT_TIME: Power sequencer; verified with WAIT_TIME=1.

23.7.13 HDMI_NV_PDISP_SOR_SEQ_INST1_0

Usage: boot / initialization

Offset: 0x61 │ Byte Offset: 0x184 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE

Bit	Reset	Description
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	0x0	PIN_B: 0 = LOW 1 = HIGH
21	0x0	PIN_A: 0 = LOW 1 = HIGH
15	0x1	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

23.7.14 HDMI_NV_PDISP_SOR_SEQ_INST2_0

Usage: boot / initialization

Offset: 0x62 │ Byte Offset: 0x188 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE

Bit	Reset	Description
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	0x0	PIN_B: 0 = LOW 1 = HIGH
21	0x0	PIN_A: 0 = LOW 1 = HIGH
15	0x1	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

23.7.15 HDMI_NV_PDISP_SOR_SEQ_INST3_0

Usage: boot / initialization

Offset: 0x63 │ Byte Offset: 0x18c │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK

Bit	Reset	Description
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	0x0	PIN_B: 0 = LOW 1 = HIGH
21	0x0	PIN_A: 0 = LOW 1 = HIGH
15	0x1	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

23.7.16 HDMI_NV_PDISP_SOR_SEQ_INST4_0

Usage: boot / initialization

Offset: 0x64 │ Byte Offset: 0x190 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE

Bit	Reset	Description
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

23.7.17 HDMI_NV_PDISP_SOR_SEQ_INST5_0

Usage: boot / initialization

Offset: 0x65 │ Byte Offset: 0x194 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE

Bit	Reset	Description
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

23.7.18 HDMI_NV_PDISP_SOR_SEQ_INST6_0

Usage: boot / initialization

Offset: 0x66 │ Byte Offset: 0x198 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	0x0	PIN_B: 0 = LOW 1 = HIGH

Bit	Reset	Description
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

23.7.19 HDMI_NV_PDISP_SOR_SEQ_INST7_0

Usage: boot / initialization

Offset: 0x67 │ Byte Offset: 0x19c │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	0x0	PIN_B: 0 = LOW 1 = HIGH
21	0x0	PIN_A: 0 = LOW 1 = HIGH

Bit	Reset	Description
15	0x1	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

23.7.20 HDMI_NV_PDISP_SOR_SEQ_INST8_0

Usage: boot / initialization

Offset: 0x68 │ Byte Offset: 0x1a0 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000) | Default: 0x00802001

Bit	Reset	Default	Description
31	0x0	NONE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	NONE	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	NONE	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	NONE	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	NONE	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	NONE	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	NONE	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	NONE	TRISTATE_IOS: 0 is the correct value for power-up. 1 is the value when HDMI is off. 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	TRUE	DRIVE_PWM_OUT_LO: Power sequencer; verified with value 1 (TRUE). 0 = FALSE 1 = TRUE
22	0x0	NONE	PIN_B: 0 = LOW 1 = HIGH
21	0x0	NONE	PIN_A: 0 = LOW 1 = HIGH
15	0x1	NONE	HALT: 0 = FALSE 1 = TRUE

Bit	Reset	Default	Description
13:12	0x0	VSYNC	WAIT_UNITS: Power sequencer; verified with WAIT_UNITS=2 (VSYNC) 0 = US 1 = MS 2 = VSYNC
9:0	0x0	0x1	WAIT_TIME: Power sequencer; verified with WAIT_TIME=1.

23.7.21 HDMI_NV_PDISP_SOR_SEQ_INST9_0

Usage: boot / initialization

Offset: 0x69 │ Byte Offset: 0x1a4 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
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

23.7.22 HDMI_NV_PDISP_SOR_SEQ_INSTA_0

Usage: boot / initialization

Offset: 0x6a │ Byte Offset: 0x1a8 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
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

23.7.23 HDMI_NV_PDISP_SOR_SEQ_INSTB_0

Usage: boot / initialization

Offset: 0x6b │ Byte Offset: 0x1ac │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
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

23.7.24 HDMI_NV_PDISP_SOR_SEQ_INSTC_0

Usage: boot / initialization

Offset: 0x6c │ Byte Offset: 0x1b0 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
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

23.7.25 HDMI_NV_PDISP_SOR_SEQ_INSTD_0

Usage: boot / initialization

Offset: 0x6d │ Byte Offset: 0x1b4 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
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

23.7.26 HDMI_NV_PDISP_SOR_SEQ_INSTE_0

Usage: boot / initialization

Offset: 0x6e │ Byte Offset: 0x1b8 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
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

23.7.27 HDMI_NV_PDISP_SOR_SEQ_INSTF_0

Usage: boot / initialization

Offset: 0x6f │ Byte Offset: 0x1bc │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK
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

23.7.28 HDMI_NV_PDISP_SOR_LANE_DRIVE_CURRENT_0

TMDS per-lane I/O current control.

Although each control is specified as 8 bits below, making it easy to set the value in hex, only the 7 LSBs of each lane are used. The mapping of binary values to current is as follows:

- 0000000: 0 mA

- 0000001: 0.4mA
- 0000010: 0.8mA
- 0000011: 1.2mA
- 0000100: 1.6mA
- 0000101: 2.0mA
- 0000110: 2.4mA
- 0000111: 2.8mA
- 0001000: 3.2mA
- 0001001: 3.6mA
- 0001010: 4.0mA
- 0001011: 4.4mA
- 0001100: 4.8mA
- 0001101: 5.2mA
- 0001110: 5.6mA
- 0001111: 6.0mA
- 0010000: 6.4mA
- 0010001: 6.8mA
- 0010010: 7.2mA
- 0010011: 7.6mA
- 0010100: 8.0mA
- 0010101: 8.4mA
- 0010110: 8.8mA
- 0010111: 9.2mA
- 0011000: 9.6mA
- 0011001: 10.0mA
- 0011010: 10.4mA
- 0011011: 10.8mA
- 0011100: 11.2mA
- 0011101: 11.6mA
- 0011110: 12.0mA
- 0011111: 12.4mA
- 0100000: 12.8mA
- 0100001: 13.2mA
- 0100010: 13.6mA
- 0100011: 14.0mA
- 0100100: 14.4mA
- 0100101: 14.8mA
- 0100110: 15.2mA
- 0100111: 15.6mA
- 0101000: 16.0mA
- 0101001: 16.4mA

- 0101010: 16.8mA
- 0101011: 17.2mA
- 0101100: 17.6mA
- 0101101: 18.0mA
- 0101110: 18.4mA
- 0101111: 18.8mA
- 0110000: 19.2mA (start to borrow pre-amp current)
- 0110001: 19.6mA
- 0110010: 20.0mA
- 0110011: 20.4mA
- 0110100: 20.8mA
- 0110101: 21.2mA
- 0110110: 21.6mA
- 0110111: 22.0mA
- 0111000: 22.4mA
- 0111001: 22.8mA
- 0111010: 23.2mA
- 0111011: 23.6mA
- 0111100: 24.0 mA
- 0111101: 24.4mA
- 0111110: 24.8mA
- 0111111: 25.2mA
- 1000000: 25.4mA
- 1000001: 25.8mA
- 1000010: 26.2mA
- 1000011: 26.6mA
- 1000100: 27.0mA
- 1000101: 27.4mA
- 1000110: 27.8mA
- 1000111: 28.2mA (maximum value)
- 1001000: 25.4mA
- 1001001: 25.8mA
- 1001010: 26.2mA
- 1001011: 26.6mA
- 1001100: 27.0mA
- 1001101: 27.4mA
- 1001110: 27.8mA
- 1001111: 28.2mA
- 1010000: 25.4mA
- ...

1111111: 28.2 mA

By default, FUSE_OVERRIDE is FALSE, meaning that the TMDS current control is driven via calibration data stored in the fuse block. Software can override these defaults by programming FUSE_OVERRIDE to TRUE, and setting each LANEn field to the appropriate value.

Offset: 0x7e │ Byte Offset: 0x1f8 │ Read/Write: R/W │ Reset: 0x20202020 (0b00100000001000000010000000100000)

Bit	Reset	Description
31:24	0x20	LANE3
23:16	0x20	LANE2
15:8	0x20	LANE1
7:0	0x20	LANE0

23.7.29 HDMI_NV_PDISP_SOR_REFCLK_0

SOR_REFCLK

The HDMI clock is programmable and varies, depending on screen resolution. The NV_PDISP_SOR_SEQ_INSTn instructions (above) can wait for certain time intervals to elapse. Whenever hdmi_clk frequency is changed, this register must be reprogrammed to divide hdmi_clk to produce a 1 µs time reference, otherwise the time intervals requested by NV_PDISP_SOR_SEQ_INSTn will not be accurate.

The format of DIVISOR is an unsigned 8.2 divider. Because this is a simple digital divider, not a PLL, fractional values result in a jitter of one hdmi_clk between successive "1 µs" intervals, but the long-term average works out to the requested divisor. This jitter is OK because the NV_PDISP_SOR_SEQ_INST wait intervals do not need to be exact. If the integer part is written as 0, it will be interpreted the same as "1".

Offset: 0x95 │ Byte Offset: 0x254 │ Read/Write: R/W │ Reset: 0x00001900 (0bxxxxxxxxxxxxxxxx0001100100xxxxxx)

Bit	Reset	Description
15:8	0x19	DIV_INT: default: 27 MHz
7:6	0x0	DIV_FRAC

23.7.30 HDMI_NV_PDISP_SOR_IO_PEAK_CURRENT_0

Pad controls for 28nm macro TMDS_X4_HP

Transmitter De-emphasis Current

The register fields are 8 bits wide but only the 7 LSBs are currently used. The mapping of register value to current is as follows:

- 0000000: 0mA
- 0000001: 0.2mA
- 0000010: 0.4mA
- 0000011: 0.6mA
- 0000100: 0.8mA
- 0000101: 1.0mA
- 0000110: 1.2mA
- 0000111: 1.4mA
- 0001000: 1.6mA
- 0001001: 1.8mA

- 0001010: 2.0mA
- 0001011: 2.2mA
- 0001100: 2.4mA
- 0001101: 2.6mA
- 0001110: 2.8mA
- 0001111: 3.0mA
- 0010000: 3.2mA
- 0010001: 3.4mA
- 0010010: 3.6mA
- 0010011: 3.8mA
- 0010100: 4.0mA
- 0010101: 4.2mA
- 0010110: 4.4mA
- 0010111: 4.6mA
- 0011000: 4.8mA
- 0011001: 5.0mA
- 0011010: 5.2mA
- 0011011: 5.4mA
- 0011100: 5.6mA
- 0011101: 5.8mA
- 0011110: 6.0mA
- 0011111: 6.2mA
- 0100000: 6.4mA
- 0100001: 6.6mA
- 0100010: 6.8mA
- 0100011: 7.0mA
- 0100100: 7.2mA
- 0100101: 7.4mA
- 0100110: 7.6mA
- 0100111: 7.8mA
- 0101000: 8.0mA
- 0101001: 8.2mA
- 0101010: 8.4mA
- 0101011: 8.6mA
- 0101100: 8.8mA
- 0101101: 9.0mA
- 0101110: 9.2mA
- 0101111: 9.4mA (maximum pre-amp current)
- 0110000: 8.0mA
- 0110001: 8.2mA
- 0110010: 8.4mA

- 0110011: 8.6mA
- 0110100: 8.8mA
- 0110101: 9.0mA
- 0110110: 9.2mA
- 0110111: 9.4mA
- 0111000: 8.0mA
- ...
- 1111111: 9.4mA

Offset: 0xd1 | Byte Offset: 0x344 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	LANE3
23:16	0x0	LANE2
15:8	0x0	LANE1
7:0	0x0	LANE0

23.7.31 HDMI_NV_PDISP_SOR_PAD_CTLS0_0

Offset: 0xd2 | Byte Offset: 0x348 | Read/Write: R/W | Reset: 0x000034bb (0b0xxxxx00x00000000011010010111011) | Default: 0x80000000

Bit	Reset	Default	Description
31	FALSE	TRUE	FUSE_OVERRIDE: 0 = FALSE 1 = TRUE (software default)
25	0x0	_NONE_	LOADADJ_SYNC_EN
24	0x0	_NONE_	LOADADJ_BYPN
22	0x0	_NONE_	REG_BYPASS
21	0x0	_NONE_	KVCO_2NDVCO
20	0x0	_NONE_	VCOCALIB_TS0
19	0x0	_NONE_	VCOCALIB_OVRWRB
18	0x0	_NONE_	VCOCALIB_ENB
17	0x0	_NONE_	VCOLIMIT_DISABLE
16	0x0	_NONE_	VCOLIMIT_SEL
15:12	0x3	_NONE_	BG_TEMP_COEF
11:8	0x4	_NONE_	BG_VREF_LEVEL
7:6	0x2	_NONE_	AVDD23_LEVEL
5:4	0x3	_NONE_	AVDD23_LOAD
3:2	0x2	_NONE_	AVDD10_LEVEL
1:0	0x3	_NONE_	AVDD10_LOAD

23.7.32 HDMI_NV_PDISP_SOR_PAD_CTLS1_0

Offset: 0xd3 | Byte Offset: 0x34c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11	0x0	DIV_RATIO_OVERRIDE
10	0x0	PLL_BYPASS
9:8	0x0	KICKSTART
7:4	0x0	PLL_NDIV_RATIO
3:0	0x0	PLL_PDIV_RATIO

23.8 Test and Debug Registers

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

23.8.1 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 that have been re-parallelized from the serial data going out.

These encoded bits of data are broken up into three 16-bit pieces depending on either LVDS or TMDS encoding:

For TMDS, the encoded data is as follows:

- CRCH: {8'b00000000,TMDS_CLK_ENC[9:2]}
- CRCM: {TMDS_CLK_ENC[1:0],TMDS_TX2_ENC[9:0],TMDS_TX1_ENC[9:6]}
- CRCL: {TMDS_TX1_ENC[5:0],TMDS_TX0_ENC[9:0]}

For LVDS, the encoded data is as follows:

- CRCH: {8'b00000000,LVDS_CLK_ENC[6:4]}
- CRCM: {LVDS_CLK_ENC[3:0],,LVDS_A3_ENC[6:0],LVDS_A2_ENC[6:2]}
- CRCL: {LVDS_A2_ENC[1:0],LVDS_A1_ENC[6:0],LVDS_A0_ENC[6:0]}

For each of these pieces, a CRC16 is computed, CRCH for the upper 16-bit piece, CRCM forms the middle 16-bit piece, and CRCL for the lower 16-bit piece. The CRC16 takes in the 16 bits of encoded data plus the CRC16 value of the previous cycle to generate the new CRC16. The CRC equation which is used is:

$x^{16} + x^{11} + x^4 + 1$.

At each VSync, the previously running CRC16 values are captured into the VCRC registers and a previous CRC value of all zeroes is fed into the CRC16 units. This means the running CRC16 is reset at each VSync.

Note:	For power saving, the VCRC is only computed when NV_PDISP_SOR_TRIG != 0. Also, it seems these are not latched at VSync but are free-running.

The VCRC registers for sub-link A

Usage: debug

Offset: 0x72 │ Byte Offset: 0x1c8 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	CRCM
15:0	X	CRCL

23.8.2 HDMI_NV_PDISP_SOR_VCRCA1_0

Usage: debug

Offset: 0x73 │ Byte Offset: 0x1cc │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	CRCH

23.8.3 HDMI_NV_PDISP_SOR_CCRCA0_0

The CCRC registers contain the CRC value for the encoded link, captured when the trigger event occurred.

The CCRC registers for sub-link A

Usage: debug

Offset: 0x74 | Byte Offset: 0x1d0 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	CRCM
15:0	X	CRCL

23.8.4 HDMI_NV_PDISP_SOR_CCRCA1_0

Usage: debug

Offset: 0x75 | Byte Offset: 0x1d4 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	CRCH

23.8.5 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 are up to 40 bits of encoded data. The encoded bits making up the data are as follows:

For TMDS, the encoded data is as follows:

```
EDATA[39:0] =
{TMDS_CLK_ENC[9:0],
TMDS_TX2_ENC[9:0],
TMDS_TX1_ENC[9:0],
TMDS_TX0_ENC[9:0]}
```

For LVDS, the encoded data is as follows:

```
EDATA[39:0] =
{5'b00000,
LVDS_CLK_ENC[6:0],
LVDS_A3_ENC[6:0],
LVDS_A2_ENC[6:0],
LVDS_A1_ENC[6:0],
LVDS_A0_ENC[6:0]}
```

The EDATA registers can be written to when any of the trigger bits for capture are set high.

The EDATA registers for sub-link A

Usage: debug

Offset: 0x76 | Byte Offset: 0x1d8 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL

23.8.6 HDMI_NV_PDISP_SOR_EDATAA1_0

Usage: debug

Offset: 0x77 | Byte Offset: 0x1dc | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VAL

23.8.7 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: 0x78 | Byte Offset: 0x1e0 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TX1
15:0	X	TX0

23.8.8 HDMI_NV_PDISP_SOR_COUNTA1_0

Usage: debug

Offset: 0x79 | Byte Offset: 0x1e4 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	TX2

23.8.9 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 are up to 40 bits of data to be encoded. The bits to be encoded which make up the data are as follows:

For TMDS, the data to be encoded are loaded as follows:

```
{TMDS_CLK_ENC[9:0], TMDS_TX2_ENC[9:0], TMDS_TX1_ENC[9:0], TMDS_TX0_ENC[9:0]}
 =
 {DEBUGA1, DEBUGA0}
```

For LVDS, the encoded data is as follows:

```
 {5'b00000, LVDS_CLK_ENC[6:0], LVDS_A3_ENC[6:0], LVDS_A2_ENC[6:0],
 LVDS_A1_ENC[6:0], LVDS_A0_ENC[6:0]}
 =
 {DEBUGA1[2:0], DEBUGA0}
```

Usage: debug

Offset: 0x7a | Byte Offset: 0x1e8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VAL: 0 = RESETV

23.8.10 HDMI_NV_PDISP_SOR_DEBUGA1_0

Usage: debug

Offset: 0x7b │ Byte Offset: 0x1ec │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VAL: 0 = RESETV

23.8.11 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 greater than the number of cycles between consecutive VSYNCs, then the trigger is not reset while it is still pending to capture data. Once the data has been captured, the trigger is reset with the following VSYNC to capture data again. If TRIG is all zeros, then the trigger is disabled and no capturing occurs.

Usage: debug

Offset: 0x7c │ Byte Offset: 0x1f0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:0	0x0	VAL: 0 = RESETV

23.8.12 HDMI_NV_PDISP_SOR_MSCHECK_0

The mode switch monitor is used for hardware debug only. To use, the test should set the CTL bit to CLEAR and then to RUN. At the end of the test, the register can be read. The various fields indicate the number of times the following conditions occurred:

- CRC enable went from false to true
- CRC enable went from true to false
- Data enable went from false to true
- Data enable went from true to false

All counts have 4 bits, so the count will clamp at 15.

Usage: debug

Offset: 0x7d │ Byte Offset: 0x1f4 │ Read/Write: R/W │ Reset: 0x80000000 (0b1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	0x1	CTL: 0 = CLEAR 1 = RUN
15:12	RO	X	DATA_ENABLE_T2F
11:8	RO	X	DATA_ENABLE_F2T
7:4	RO	X	CRC_ENABLE_T2F
3:0	RO	X	CRC_ENABLE_F2T

23.9 Audio Registers

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

23.9.1 HDMI_NV_PDISP_AUDIO_N_0

AUDIO_N

N_VALUE: N_VALUE is the N parameter in HDMI Audio Clock Regeneration Packet. This value should be written when hardware measured CTS is enabled. The correct N value can be read from tables 7-1, 7-2, and 7-3 in the HDMI specification.

N_RESET: N_RESET is the reset for the N counter. If software is controlling the value of N, every time the audio stream changes sampling frequency (fs), software (driver) need to reset the N counter by writing _ASSERT to N_RESET followed by writing a _DEASSERT to N_RESET.

If the hardware selected N feature is enabled (N_LOOKUP = _ENABLE), software only needs to reset the N counter when writing to the AUDIO_NVAL registers and when enabling/disabling N_LOOKUP.

1) Set N_RESET = _ASSERT

2) Modify N value registers

3) Set N_RESET = _DEASSERT

Modifying N_VALUE can be done in step 1.

N_GENERATE: N_GENERATE controls how the audio block generates the 128*fs/N pulse, which is related to CTS. The detail of CTS can be found in HDMI 1.1 specification, Chapter 7. This bit is strictly for debugging purposes only.

_NORMAL: This method increments the CTS counter a variable number of times based on the length of the S/PDIF pulse.

_ALTERNATE: This method attempts to recreate the 128*fs clock and increments the CTS counter at regular intervals based on HALF.

N_LOOKUP: When set to _ENABLE, the hardware will select the appropriate value of N to use from one of the AUDIO_NVAL registers. This selection is based on the audio sampling frequency detected by the audio block. This is not the sampling frequency reported in the channel status bits. N_RESET should be toggled when this feature is enabled. When set to _DISABLE, software must program the correct N value into AUDIO_N.

Offset: 0x8c │ Byte Offset: 0x230 │ Read/Write: R/W │ Reset: 0x01000000 (0bxxx0xxx1xxx00000000000000000000)

Bit	Reset	Description
28	0x0	LOOKUP: 1 = ENABLE 0 = DISABLE
24	0x1	GENERATE: 0 = NORMAL 1 = ALTERNATE
20	0x0	RESETF: 1 = ASSERT 0 = DEASSERT
19:0	0x0	VALUE

23.9.2 HDMI_NV_PDISP_SOR_AUDIO_CNTRL0_0

HD Audio (also known as Azalia)

PORT_CONNECTIVITY: This controls the behavior of the Port Connectivity field of the Azalia Configuration Defaults Verb. This can be used to disable a codec that is associated with an SOR that does not connect to a physical port.

- ENABLE: Report 00 in the Port Connectivity field by default. This corresponds to "The Port Complex is connected to a jack"
- DISABLE: Report 01 in the Port Connectivity field by default. This corresponds to "No physical connection for Port" See table 100 and table 101 of the High Definition Audio Specification (Revision 1.0) for more information.

AFIFO_FLUSH: When the DP and HDMI logic are not in the middle of sending an audio packet, they will flush out any entries at the head of the AFIFO that is not tagged as the beginning of a sample. This ensures that the next new audio packet sent will begin on the correct channel. This is a diagnostic workaround bit to disable the flushing.

- ENABLE: Automatically fix the AFIFO if it gets out of alignment
- DISABLE: Do not throw out any AFIFO entries.

SAMPLING_FREQ: This will report the incoming audio stream sampling frequency in the Azalia codec. The HDMI specification only supports 7 audio sample frequencies (fs). Anything other than those 7 fs will be reported as UNKNOWN. See the HDMI 1.1 specification, Table 7-4 (page 77).

SOURCE_SELECT: Determines whether to use the S/PDIF or Azalia (HDAL) audio input. This field has no effect on chips without the HDAL input. The default is AUTO: selects S/PDIF audio until the HDAL audio input is initialized by the external controller

INJECT_NULLSMPL: When the bit is enabled, if the audio format is stereo LPCM as indicated by the stream format in the corresponding output converter widget, the codec inserts null samples into the audio FIFO for each Azalia frame in which it did not receive any samples. This is done only for stereo LPCM and not for any other audio format. This bit should be disabled by default for backwards compatibility.

INPUT_MODE: This bit indicates what the audio data source is. HDA is Azalia data, S/PDIF is S/PDIF data.

SOR_AUDIO_CNTRL0

Offset: 0xac │ Byte Offset: 0x2b0 │ Read/Write: R/W │ Reset: 0xX01X1000 (0bxx0xxxxxxx01xxxxxxx1xxxxxxxxxxx0)

Bit	R/W	Reset	Description
31	RO	X	INPUT_MODE: 0 = HDA 1 = SPDIF
29	RW	DISABLE	INJECT_NULLSMPL: 0 = DISABLE 1 = ENABLE
21:20	RW	SPDIF	SOURCE_SELECT: 0 = AUTO 1 = SPDIF 2 = HDAL
19:16	RO	X	SAMPLING_FREQ: 3 = FREQ_32_0KHZ 0 = FREQ_44_1KHZ 8 = FREQ_88_2KHZ 12 = FREQ_176_4KHZ 2 = FREQ_48_0KHZ 10 = FREQ_96_0KHZ 14 = FREQ_192_0KHZ 1 = FREQ_UNKNOWN

Bit	R/W	Reset	Description
12	RW	ENABLED	AFIFO_FLUSH: 0 = DISABLED 1 = ENABLED
0	RW	ENABLE	PORT_CONNECTIVITY: 0 = ENABLE 1 = DISABLE

23.9.3 HDMI_NV_PDISP_SOR_AUDIO_DEBUG_0

SOR_AUDIO_DEBUG

This register is used for debug purposes only.

FIFO_ERROR: The data is passed from the audio (Azalia or S/PDIF) to the SOR via an async FIFO. This bit indicates that the FIFO is full. There is one bit for each head. This is an indication that the SOR units do not drain away the audio sample data fast enough.

Offset: 0xad │ Byte Offset: 0x2b4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	NO	FIFO_ERROR: 0 = NO 1 = YES

23.9.4 HDMI_NV_PDISP_SOR_AUDIO_SPARE0_0

This register is a backup register.

SOR_AUDIO_SPARE0

Offset: 0xae │ Byte Offset: 0x2b8 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	Reserved.

23.9.5 HDMI_NV_PDISP_SOR_AUDIO_NVAL_0320_0

SOR_AUDIO_NVAL

The following seven registers are used for HDMI N values for Azalia. These registers should be written with the correct N values for the current pixel clock frequency. These values can be found in tables 7-1, 7-2, and 7-3 of the HDMI specification. These registers are initialized to the value in the "other" row of these tables.

For DisplayPort audio, a value of 2^15 (0x8000) is always used.

VALUE: The correct N value for 32 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xaf │ Byte Offset: 0x2bc │ Read/Write: R/W │ Reset: 0x00001000 (0bxxxxxxxxxxxxx00000001000000000000)

Bit	Reset	Description
19:0	0x1000	VALUE

23.9.6 HDMI_NV_PDISP_SOR_AUDIO_NVAL_0441_0

VALUE: The correct N value for 44.1 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xb0 | Byte Offset: 0x2c0 | Read/Write: R/W | Reset: 0x00001880 (0bxxxxxxxxxxxx00000011000010000000)

Bit	Reset	Description
19:0	0x1880	VALUE

23.9.7 HDMI_NV_PDISP_SOR_AUDIO_NVAL_0882_0

VALUE: The correct N value for 88.2 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xb1 | Byte Offset: 0x2c4 | Read/Write: R/W | Reset: 0x00003100 (0bxxxxxxxxxxxx00000011000100000000)

Bit	Reset	Description
19:0	0x3100	VALUE

23.9.8 HDMI_NV_PDISP_SOR_AUDIO_NVAL_1764_0

VALUE: The correct N value for 176.4 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xb2 | Byte Offset: 0x2c8 | Read/Write: R/W | Reset: 0x00006200 (0bxxxxxxxxxxxx00000110001000000000)

Bit	Reset	Description
19:0	0x6200	VALUE

23.9.9 HDMI_NV_PDISP_SOR_AUDIO_NVAL_0480_0

VALUE: The correct N value for 48 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xb3 | Byte Offset: 0x2cc | Read/Write: R/W | Reset: 0x00001800 (0bxxxxxxxxxxxx00000011000000000000)

Bit	Reset	Description
19:0	0x1800	VALUE

23.9.10 HDMI_NV_PDISP_SOR_AUDIO_NVAL_0960_0

VALUE: The correct N value for 96 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xb4 | Byte Offset: 0x2d0 | Read/Write: R/W | Reset: 0x00003000 (0bxxxxxxxxxxxx00000011000000000000)

Bit	Reset	Description
19:0	0x3000	VALUE

23.9.11 HDMI_NV_PDISP_SOR_AUDIO_NVAL_1920_0

VALUE: The correct N value for 192 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xb5 | Byte Offset: 0x2d4 | Read/Write: R/W | Reset: 0x00006000 (0bxxxxxxxxxxxx00000110000000000000)

Bit	Reset	Description
19:0	0x6000	VALUE

23.9.12 HDMI_NV_PDISP_SOR_AUDIO_HDA_SCRATCH0_0

SOR_AUDIO_HDA_SCRATCH0/1/2/3

This is a field to be used if it is determined at a later time that additional information needs to be sent from the display driver to the audio driver for support of any of the extended formats.

Offset: 0xb6 | Byte Offset: 0x2d8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RESERVED

23.9.13 HDMI_NV_PDISP_SOR_AUDIO_HDA_SCRATCH1_0

Offset: 0xb7 | Byte Offset: 0x2dc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RESERVED

23.9.14 HDMI_NV_PDISP_SOR_AUDIO_HDA_SCRATCH2_0

Offset: 0xb8 | Byte Offset: 0x2e0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RESERVED

23.9.15 HDMI_NV_PDISP_SOR_AUDIO_HDA_SCRATCH3_0

Offset: 0xb9 | Byte Offset: 0x2e4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RESERVED

23.9.16 HDMI_NV_PDISP_SOR_AUDIO_HDA_CODEC_SCRATCH0_0

These registers are set by the audio driver using vendor-defined verbs. They can be used to pass information from the audio driver to the resource manager if that functionality is ever needed.

When bit 31 changes, an interrupt can be generated (see INT_STATUS register)

Offset: 0xba | Byte Offset: 0x2e8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

23.9.17 HDMI_NV_PDISP_SOR_AUDIO_HDA_CODEC_SCRATCH1_0

Offset: 0xbb | Byte Offset: 0x2ec | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

23.9.18 HDMI_NV_PDISP_SOR_AUDIO_HDA_ELD_BUFWR_0

Software Programming Model

In an integrated graphics chip, which includes an NVIDIA® Audio Controller, with support for an NVIDIA audio codec driver, the following is the background and the suggested programming model.

The audio software for the HDMI codec will need information about the audio capabilities of an attached HDMI sink device. This information is stored in the HDMI sink device's EDID. Typically, the EDID flows through a graphics adapter to graphics software, so the graphics adapter hardware will not have knowledge of the EDID contents.

To that end, a new mechanism is defined for passing the HDMI sink device's audio EDID information from the graphics software to the audio software. The data payload containing the audio information will be known as EDID-Like Data (or ELD) and will contain a subset of the HDMI sink devices EDID information.

The ELD information will be valid if the HDMI sink is attached and powered on and the ELD Valid bit is set. The Pin Widget that is associated with this HDMI widget will report if the device is attached and that the ELD memory is populated and valid by reporting Presence Detect of 1 and ELD Valid of 1 to a Pin Sense control command. As with the Presence Detect bit, the changes to the ELD Valid bit can also result in the generation of unsolicited responses.

Each codec implements a 96-byte ELD buffer that is written by the resource manager and read by the audio driver. Once the ELD buffer is written by the resource manager, the valid bit, NV_PDISP_SOR_AUDIO_HDA_PRESENSE_ELDV is set to _VALID to indicate that ELD contents have been initialized by the resource manager. On hot unplug events, NV_PDISP_SOR_AUDIO_HDA_PRESENSE_ELDV is set to _INVALID to erase ELD programming in the audio driver.

Offset: 0xbc │ Byte Offset: 0x2f0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	INDEX
7:0	0x0	DATABYTE

23.9.19 HDMI_NV_PDISP_SOR_AUDIO_HDA_PRESENSE_0

Reports the Hot Plug state and ELD state to the audio driver. Changes to this state can cause an unsolicited response.

ELDV: Indicates whether the data in the ELD buffer is valid and ready to read.

PD: Presence Detect. This should be set to _PRESENT by software upon hot plug and _NOT_PRESENT on unplug.

Offset: 0xbd │ Byte Offset: 0x2f4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	INVALID	ELDV: 0 = INVALID 1 = VALID
0	NOT_PRESENT	PD: 0 = NOT_PRESENT 1 = PRESENT

23.9.20 HDMI_NV_PDISP_SOR_AUDIO_HDA_CP_0

Reports the Content Protection state requested by the Audio driver. It is at the discretion of the video driver to enable or disable content protection. This is a read-only register set by the Content Protection Control (CP_CONTROL) verb.

REQUEST_STATE

_DONT_CARE: No state change requested

_RESERVED: Unused

_PROTECTION_OFF: Audio driver requests content protection to be disabled.

_PROTECTION_ON: Audio driver requests content protection to be enabled.

Offset: 0xbe │ Byte Offset: 0x2f8 │ Read/Write: RO │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2	X	REQUEST_STATE_VALID
1:0	X	REQUEST_STATE:

Bit	Reset	Description
		0 = DONT_CARE 2 = PROTECTION_OFF 3 = PROTECTION_ON

23.9.21 HDMI_NV_PDISP_SOR_AUDIO_AVAL_0320_0

When using the Azalia codec, it is difficult to recover the 128*fs clock frequency from the incoming audio stream. Therefore, when using the Azalia codec, these registers must be programmed to emulate the N counter.

When using S/PDIF, the N counter will run at 128*fs (audio sampling frequency) and count to a value determined by the HDMI specification. The Azalia counter will run at 24 MHz at all times, and it needs to count for the same period of time as the N counter would have.

Refer to section 7.2 of the HDMI specification, 1.2.

AVAL values do not need to be programmed by software. Their default values are OK. If the NVAL changes, it will change the N counter frequency. AVAL will need to be changed to match with this new NVAL. Certain resolutions may require different NVALs.

For Mobile chips, the clock is 24 MHz, not 54 MHz. For 44.1, the default N of 6272 gives non-integer AVAL. Choosing N of 4704 instead (nominal CTS 61875) gives an N frequency of 1200 Hz.

48k: 1ms 24000 azaclk cycles (0x5DC0)

44.1k: 1ms/1.2 20000 azaclk cycles (0x4E20)

But this is academic because CTS/N is hardcoded since the pixel clock / audio clock ratios are known and fixed.

At 27 MHz Pixel Clock			
	CTS	**N**	**AVAL**
44.1	22500	4704	20000
88.2	22500	9408	20000
176.4	22500	18816	20000
At 74.25 MHz Pixel Clock			
44.1	61875	4704	20000
88.2	61875	9408	20000
176.4	61875	18816	20000
At 148.5 MHz Pixel Clock:			
44.1	123750	4704	20000
88.2	123750	9408	20000
176.4	123750	18816	20000

VALUE: The correct A value for 32 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xbf │ Byte Offset: 0x2fc │ Read/Write: R/W │ Reset: 0x00005dc0 (0bxxxxxxxxxxxxx0000010111011100000 0)

Bit	Reset	Description
19:0	0x5dc0	VALUE

23.9.22 HDMI_NV_PDISP_SOR_AUDIO_AVAL_0441_0

VALUE: The correct A value for 44.1 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xc0 │ Byte Offset: 0x300 │ Read/Write: R/W │ Reset: 0x00004e20 (0bxxxxxxxxxxxxx0000010011100010 0000)

Bit	Reset	Description
19:0	0x4e20	VALUE

23.9.23 HDMI_NV_PDISP_SOR_AUDIO_AVAL_0882_0

VALUE: The correct A value for 88.2 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xc1 | Byte Offset: 0x304 | Read/Write: R/W | Reset: 0x00004e20 (0bxxxxxxxxxxxx00000100111000100000)

Bit	Reset	Description
19:0	0x4e20	VALUE

23.9.24 HDMI_NV_PDISP_SOR_AUDIO_AVAL_1764_0

VALUE: The correct A value for 176.4 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xc2 | Byte Offset: 0x308 | Read/Write: R/W | Reset: 0x00004e20 (0bxxxxxxxxxxxx00000100111000100000)

Bit	Reset	Description
19:0	0x4e20	VALUE

23.9.25 HDMI_NV_PDISP_SOR_AUDIO_AVAL_0480_0

VALUE: The correct A value for 48 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xc3 | Byte Offset: 0x30c | Read/Write: R/W | Reset: 0x00005dc0 (0bxxxxxxxxxxxx00000101110111000000)

Bit	Reset	Description
19:0	0x5dc0	VALUE

23.9.26 HDMI_NV_PDISP_SOR_AUDIO_AVAL_0960_0

VALUE: The correct A value for 96 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xc4 | Byte Offset: 0x310 | Read/Write: R/W | Reset: 0x00005dc0 (0bxxxxxxxxxxxx00000101110111000000)

Bit	Reset	Description
19:0	0x5dc0	VALUE

23.9.27 HDMI_NV_PDISP_SOR_AUDIO_AVAL_1920_0

VALUE: The correct A value for 192 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xc5 | Byte Offset: 0x314 | Read/Write: R/W | Reset: 0x00005dc0 (0bxxxxxxxxxxxx00000101110111000000)

Bit	Reset	Description
19:0	0x5dc0	VALUE

23.9.28 HDMI_NV_PDISP_SOR_AUDIO_AVAL_DEFAULT_0

VALUE: Default A value if the Azalia codec sampling frequency does not match any of the above.

Offset: 0xc6 | Byte Offset: 0x318 | Read/Write: R/W | Reset: 0x00005dc0 (0bxxxxxxxxxxxx00000101110111000000)

Bit	Reset	Description
19:0	0x5dc0	VALUE

23.9.29 HDMI_NV_PDISP_SOR_AUDIO_GEN_CTRL_0

This register only takes effect when it is written by software. The initialized value is not used.

DEV_ID: Device ID to identify the current chip.

REV_ID: Rev ID for the codec. This value is only used by the codec once this register has been written. Otherwise the default value will be used.

Offset: 0xc7 │ Byte Offset: 0x31c │ Read/Write: R/W │ Reset: 0x00280001 (0b00000000000101000xxxxxxx00000001)

Bit	Reset	Description
31:16	0x28	DEV_ID
7:0	0x1	REV_ID

23.9.30 HDMI_NV_HDACODEC_AUDIO_GEN_CTL_0

CHSTS_FS_3840

Currently the 4-bit coding for 384 kHz sampling rate is not available in the IEC-61937 specification. If the specification defines this value later, software needs to write the value during device initialization.

COPY_POLARITY: The polarity of the COPY bit is currently inverted in hardware as done in previous Tegra devices.

Offset: 0xd5 │ Byte Offset: 0x354 │ Read/Write: R/W │ Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx01111)

Bit	Reset	Description
4	OLD	COPY_POLARITY: 0 = OLD 1 = NEW
3:0	0xf	CHSTS_FS_3840

[THIS PAGE INTENTIONALLY LEFT BLANK]

24.0 LVDS/EDP DISPLAY OUTPUT

24.1 Overview

The Tegra® K1 LVDS/eDP Serial Output Resource (SOR) block is a display output controller, with two operating modes, supporting either Low Voltage Differential Signaling (LVDS) or Embedded DisplayPort (eDP). It drives local panels only and does not support an external DP port.

The LVDS/eDP SOR block collects pixels from the output of the display pipeline, formats/encodes them to the LVDS/eDP format, and then streams them to various output devices. The LVDS/eDP block consists of individual resources that can interface with different display devices. An LVDS/eDP block can drive only a single device at any given time and can be connected to any display controller (Head).

24.2 Features

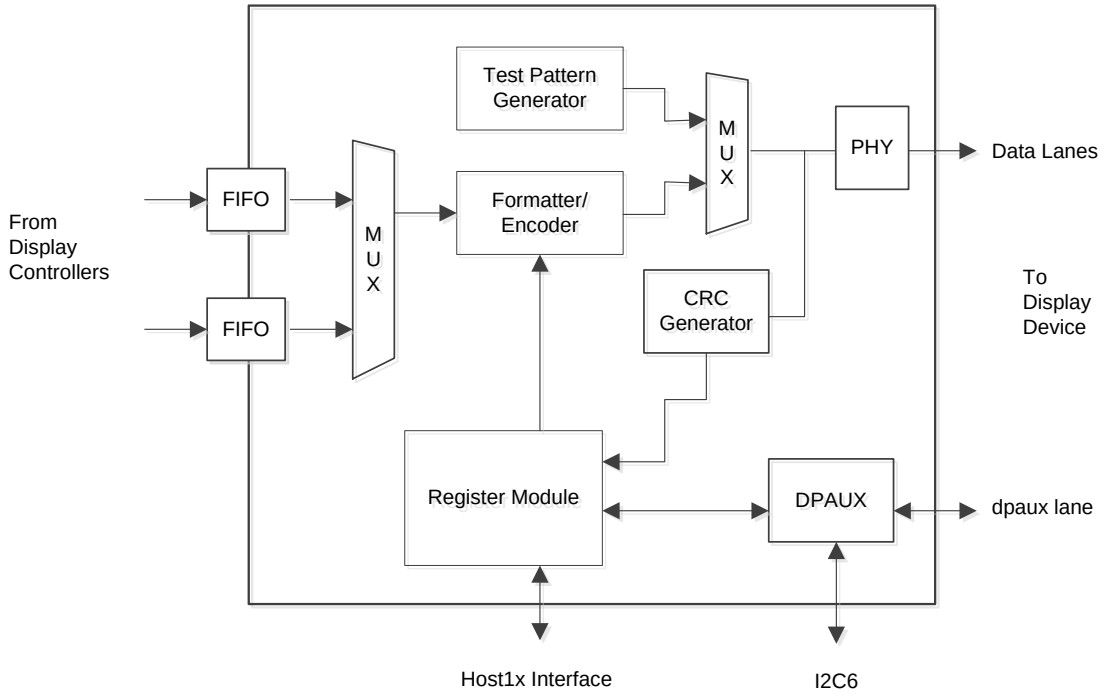
These features are supported in the LVDS/eDP SOR block:

- eDP 1.4
  - 1/2/4 lanes, single link
  - RBR/HBR/HBR2
  - 18/24 bits color depth
  - Up to 450MHz
  - Internal panel:
    - 3200 x 2000 @ 60Hz (2D – portrait/landscape)
    - 1920 x 1080 @ 60Hz (3D – portrait/landscape)
  - -0.5% down spread support
- LVDS
  - 3/4 data lanes + 1 clock lane, single link
  - 18/24 bits color depth
  - Intel 24.0/24.1 single channel format
  - Up to 165 MHz
  - Internal panel:
    - 1920 x 1200 @ 60Hz (2D – portrait/landscape)
  - LVDS direct connect (captive panel)
  - Spread-spectrum capable to -5.0%
  - Additional I2C port for LVDS
- Stereo mode
- Generic infoframe
- Supports DP AUX and HPD
- Single PHY to select LVDS or eDP as output
  - Either a 4-lane eDP or a 5-lane (single-link) LVDS
  - Supports 5.4 GHz eDP or 165MHz pixel clock LVDS

24.3 Functional Description

The LVDS/eDP block has a pixel bus as an input from the display pipe of each head. It also has a small test pattern generator and formatter to encode pixels for specific output devices, a CRC generator, and a bundle interface to the display software interface.

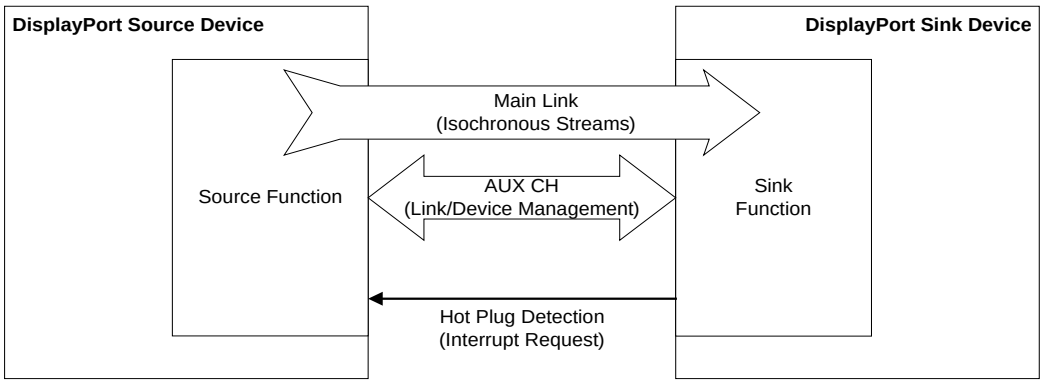
Figure 81: LVDS/eDP High-Level Block Diagram



24.4 DisplayPort Overview

The DisplayPort link consists of Main Link, Auxiliary Channel (AUX CH), and Hot Plug Detect (HPD) channels. Main Link is a unidirectional, high-bandwidth, and low-latency channel used for transport of isochronous streams such as uncompressed video and audio. Auxiliary Channel is a half-duplex, bidirectional channel used for link management and device control. The HPD signal serves as an interrupt request by the sink device. Main Link consists of AC-coupled, doubly terminated differential pairs (called lanes). Three link rates are supported: 5.4 Gbps, 2.7 Gbps, and 1.62 Gbps per lane. The Main Link supports 1, 2, or 4 lanes.

Figure 82: DisplayPort Link



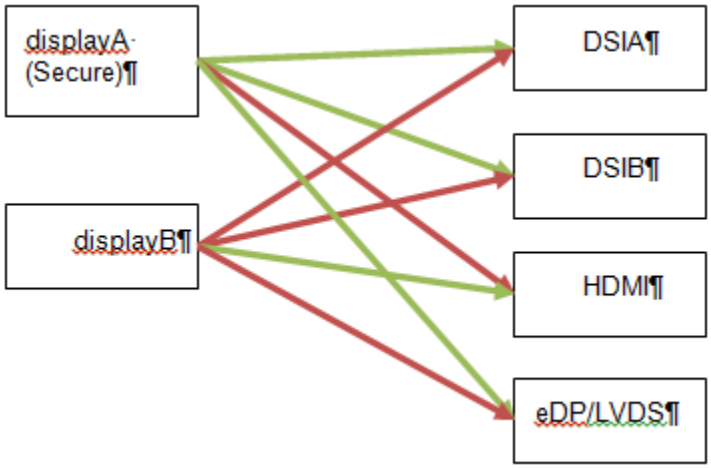
24.5 SOR Security

The eDP/LVDS SOR block does not support TZ_SECURE access.

The ARM TrustZone document requires Secure OS to control the connection of Secure Window to displays.

The typical Secure Display use case is below; green arrows are permitted SOR connections, and red arrows are disabled. All paths can be controlled by the Secure OS using the SECURE_CONTROL registers.

Figure 83: Secure Display Use Case



The TrustZone window will drive an interrupt similar to underflow interrupts from other windows. Mask and status registers related to this window are non-secure.

24.6 LVDS/eDP Registers

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

24.6.1 SOR_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: 0x0 | Byte Offset: 0x0 | Read/Write: R/W | Reset: 0xXXXXf800 (0bxxxxxxxxxxxxxxxx11111x0000000000)

Bit	R/W	Reset	Description
31:28	RO	X	NEXT_CHANNEL: Next requested channel
25:16	RO	X	NEXT_CLASS: Next requested class
15:12	RW	0xf	CURR_CHANNEL: Current working channel, reset to 'invalid'
11	RW	0x1	AUTO_ACK: Automatically acknowledge any incoming context switch requests 0 = MANUAL 1 = AUTOACK
9:0	RW	0x0	CURR_CLASS: Current working class

24.6.2 SOR_NV_PDISP_SOR_SUPER_STATE0_0

The following 2 registers are the equivalent for supervisor methods that GPU HW automatically used to enqueue.

Writing a 1 to this field cause a 1 cycle pulse which is used to activate the fields in registers NV_PDISP_SOR_STATE[1].

Offset: 0x1 | Byte Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	UPDATE

24.6.3 SOR_NV_PDISP_SOR_SUPER_STATE1_0

NV_PDISP_SOR_SUPER_STATE1 contains triple buffered registers. The values written to SUPER_STATE1 are assembly values.

These are promoted to ARMed state when SUPER_STATE0 UPDATE is written.

The ARMed values are then promoted to ACTIVE internally by HW when an internally generated LOADV signal arrives.

The active state is reported in DISP_SOR_PWR and DISP_SOR_TEST registers

ATTACHED

- YES : Attach SOR to a display head
- NO : Detach SOR from display head

ASY_ORMODE

- SAFE: SOR is in safe low power state and not sending any active data.
- NORMAL : SOR is actively sending data

ASY_HEAD_OPMODE

- SLEEP : Display is not sending pixels and SOR will stop reading pixels from FIFOs
- SNOOZE : This should never be set by SW.
- AWAKE : Display is sending active pixels to SOR

Offset: 0x2 | Byte Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	NO	ATTACHED: 0 = NO

Bit	Reset	Description
		1 = YES
2	SAFE	ASY_ORMODE: 0 = SAFE 1 = NORMAL
1:0	0x0	ASY_HEAD_OPMODE: 0 = SLEEP 1 = SNOOZE 2 = AWAKE

24.6.4 SOR_NV_PDISP_SOR_STATE0_0

Writing a 1 to this field cause a 1 cycle pulse which is used to promote the activate the fields in registers NV_PDISP_SOR_STATE[1], NV_PDISP_HEAD_STATE[0-5].

Offset: 0x3 | Byte Offset: 0xc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	UPDATE

24.6.5 SOR_NV_PDISP_SOR_STATE1_0

SOR Control Register

ASY_PIXELDEPTH

This sets the pixel depth and the only valid values are BPP_18_444 and BPP_24_444.

ASY_REPLICATE

This is used to enable/disable pixel replication for HDMI. This should always be set to OFF for Tegra as SOR0 does not support HDMI.

ASY_DEPOL

This field needs to be set based on the DE polarity as needed by the panel

ASY_VSYNCPOL

This field needs to be set based on the VSYNC polarity as needed by the panel

ASY_HSYNCPOL

This field needs to be set based on the HSYNC polarity as needed by the panel

ASY_CRCMODE

This field controls the pixels (LVDS) or symbols (eDP) that get CRCed.

- ACTIVE_RASTER --> Only active regions of the raster are CRCed
- COMPLETE_RASTER--> Entire raster (active + blanking) will be CRCed.
- NON_ACTIVE_RASTER --> Only non-active regions of the raster are CRCed

ASY_SUBOWNER

This field should always set to NONE.

ASY_OWNER

This field controls the display pipe that is being connected to the SOR. Note that the ARMED value of this filed is promoted only when SOR gets attached.

- HEAD0 --> When SOR needs to be connected to display internal head : displaya
- HEAD1 --> When SOR needs to be connected to display external head : displayb
- NONE --> When it is not connected to any heads

Offset: 0x4 | Byte Offset: 0x10 | Read/Write: R/W | Reset: 0x00000040 (0bxxxxxxxxxxxx00000000000000001000000)

Bit	Reset	Description
20:17	DEFAULTVAL	ASY_PIXELDEPTH: 0 = DEFAULTVAL 1 = BPP_16_422 2 = BPP_18_444 3 = BPP_20_422 4 = BPP_24_422 5 = BPP_24_444 6 = BPP_30_444 7 = BPP_32_422 8 = BPP_36_444 9 = BPP_48_444
16:15	OFF	ASY_REPLICATE: 0 = OFF 1 = X2 2 = X4
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	LVDS_CUSTOM	ASY_PROTOCOL: 0 = LVDS_CUSTOM 8 = DP_A 9 = DP_B 15 = CUSTOM
7:6	COMPLETE_RASTER	ASY_CRCMODE: 0 = ACTIVE_RASTER 1 = COMPLETE_RASTER 2 = NON_ACTIVE_RASTER
5:4	NONE	ASY_SUBOWNER: 0 = NONE 1 = SUBHEAD0 2 = SUBHEAD1 3 = BOTH
3:0	NONE	ASY_OWNER: 0 = NONE 1 = HEAD0 2 = HEAD1

24.6.6 SOR_NV_PDISP_HEAD_STATE0_0

Head Control Register

INTERLACED

Should always be set to PROGRESSIVE as INTERLACED is not supported

RANGECOMPRESS

Compresses the range of an RGB signal with black at 0, white at nominal 255 to the range required for CEA ranged RGB output, i.e., black at 16, white at 235. This control gives the driver a very quick way to compress RGB levels to be compatible with those required for output to TV via the TV encoder and/or to HDMI.

DYNRANGE

Sets the dynamic range for the output. VESA is the full 0 to maximum range. CEA mode will clip the output to fit within CEA range. Display Port has a restriction that BPP_18_444 RGB mode must use VESA.

COLORSPACE

Should always be set to RGB. YUV_601/YUV_709 are not supported

This is an array of 2 identical register entries; the register fields below apply to each entry.

Offset: 0x5..0x6 | Byte Offset: 0x14..0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5:4	PROGRESSIVE	INTERLACED: 0 = PROGRESSIVE 1 = INTERLACED
3	DISABLE	RANGECOMPRESS: 0 = DISABLE 1 = ENABLE
2	VESA	DYNRANGE: 0 = VESA 1 = CEA
1:0	RGB	COLORSPACE: 0 = RGB 1 = YUV_601 2 = YUV_709

24.6.7 SOR_NV_PDISP_HEAD_STATE1_0

This register sets the size of the raster

VTOTAL

This field is the total number of lines in a frame of the raster.

HTOTAL

This field is the total number of pixels in a line of the raster.

This is an array of 2 identical register entries; the register fields below apply to each entry.

Offset: 0x7..0x8 | Byte Offset: 0x1c..0x20 | Read/Write: R/W | Reset: 0x01011000 (0bx000000100000001x001000000000000)

Bit	Reset	Description
30:16	0x101	VTOTAL
14:0	0x1000	HTOTAL

24.6.8 SOR_NV_PDISP_HEAD_STATE2_0

This register sets the location of the horizontal and vertical sync end

VSYNC_END

This field is the total number of lines after which vertical sync ends

HSYNC_END

This field is the total number of pixels after which horizontal sync ends

This is an array of 2 identical register entries; the register fields below apply to each entry.

Offset: 0x9..0xa | Byte Offset: 0x24..0x28 | Read/Write: R/W | Reset: 0x00000001 (0bx0000000000000000x0000000000000001)

Bit	Reset	Description
30:16	0x0	VSYNC_END
14:0	0x1	HSYNC_END

24.6.9 SOR_NV_PDISP_HEAD_STATE3_0

This register sets the location of horizontal and vertical blank end.

VSYNC_END

This field is the total number of lines after which vertical sync ends

HSYNC_END

This field is the total number of pixels after which horizontal sync ends

This is an array of 2 identical register entries; the register fields below apply to each entry.

Offset: 0xb..0xc | Byte Offset: 0x2c..0x30 | Read/Write: R/W | Reset: 0x00010011 (0bx0000000000000001x0000000000010001)

Bit	Reset	Description
30:16	0x1	VBLANK_END
14:0	0x11	HBLANK_END

24.6.10 SOR_NV_PDISP_HEAD_STATE4_0

This register sets the location of the horizontal and vertical blank start.

This is an array of 2 identical register entries; the register fields below apply to each entry.

Offset: 0xd..0xe | Byte Offset: 0x34..0x38 | Read/Write: R/W | Reset: 0x00110100 (0bx0000000000010001x0000000100000000)

Bit	Reset	Description
30:16	0x11	VBLANK_START
14:0	0x100	HBLANK_START

24.6.11 SOR_NV_PDISP_HEAD_STATE5_0

The following fields need to be set only for interlaced mode

This is an array of 2 identical register entries; the register fields below apply to each entry.

Offset: 0xf..0x10 | Byte Offset: 0x3c..0x40 | Read/Write: R/W | Reset: 0x00000001 (0bx0000000000000000x0000000000000001)

Bit	Reset	Description
30:16	0x0	VBLANK_END_2
14:0	0x1	VBLANK_START_2

24.6.12 SOR_NV_PDISP_SOR_CRC_CNTRL_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: 0x11 | Byte Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	NO	ARM_CRC_ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

24.6.13 SOR_NV_PDISP_SOR_DP_DEBUG_MVID_0

This register reports the MVID value calculated by HW

Offset: 0x12 | Byte Offset: 0x48 | Read/Write: RO | Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:0	X	VALUE

24.6.14 SOR_NV_PDISP_SOR_CLK_CNTRL_0

LINK_SPEED

Programs the link speed for DP and LVDS i.e. it selects the multiplier used by the analog macro PLL.

For DP and TMDS, logic in the SOR will receive a version of this clock that is divided by 10. For LVDS, logic in the SOR will receive a version of this clock that is divided by 7. This field should only be changed when MODE_BYPASS is set to DP_SAFE or when the SOR is in safe mode.

It may take up to 200 microseconds for the PLLs in the analog macro to settle after this setting is changed.

Changing this field will require DP link re-training.

- G1_62: Selects the 1.62GHz link clock; PLL generates 1.62 GHz from 270MHz (multiplier = 6)
- G2_7 : Selects the 2.70GHz link clock; PLL generates 2.70GHz from 270MHz (multiplier = 10)
- G5_4 : Selects the 5.40GHz link clock; PLL generates 5.40GHz from 270MHz (multiplier = 20)
- _LVDS: Link speed is 7*pixel clock used for LVDS (multiplier = 7)

CLK_SEL

The DP_SINGLE_LVDS macro has four input clocks. This field selects which clock is used for the internal logic. When the SOR is operating in DP mode, this should be set to one of the _DPCLK settings. When the SOR is operating in LVDS mode, this should be set to one of the _PCLK settings. This field should only be changed when the SOR is asleep or the

Please allow 200 microseconds for the PLLs in the macro to settle after changing this setting.

DIFF_PCLK and DIFF_PCLK inputs to macro are tied off. So SW should only set to _SINGLE_(DP|P)CLK

- _SINGLE_PCLK: Single ended pclk from the VPLLs

- _DIFF_PCLK: Differential Pclk
- _SINGLE_DPCLK: Single ended 270MHz clock
- _DIFF_DPCLK: Differential 270MHz clock

Offset: 0x13 | Byte Offset: 0x4c | Read/Write: R/W | Reset: 0x00000018 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0011000)

Bit	Reset	Description
6:2	G1_62	DP_LINK_SPEED: 6 = G1_62 10 = G2_7 7 = LVDS 20 = G5_4
1:0	SINGLE_PCLK	DP_CLK_SEL: 0 = SINGLE_PCLK 1 = DIFF_PCLK 2 = SINGLE_DPCLK 3 = DIFF_DPCLK

24.6.15 SOR_NV_PDISP_SOR_CAP_0

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, and 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)

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 hardware. 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: 0x14 | Byte Offset: 0x50 | Read/Write: RO | Reset: 0xXXXXXX0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	LVDS_ONLY: 0 = FALSE

Bit	Reset	Description
		1 = TRUE
25	X	DP_B: 0 = FALSE 1 = TRUE
24	X	DP_A: 0 = FALSE 1 = TRUE
20	X	DDI: 0 = FALSE 1 = TRUE
16	X	SDI: 0 = FALSE 1 = TRUE
13	X	DISPLAY_OVER_PCIE: 0 = FALSE 1 = TRUE
12	X	SINGLE_TMDS_225_MHZ: 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

24.6.16 SOR_NV_PDISP_SOR_PWR_0

This register contains bits that control the power state of the SOR. For simplicity, the sequencer will be used to perform all power control operations in the SOR (even in TMDS where the sequencing is relatively simple). This unifies the approach and makes it easier for the hardware and software to deal with the SORs. 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, software should load in the new value and write SETTING_NEW=TRIGGER. Once the change has been successfully completed, the SEQ_CTL_STATUS will indicate STOPPED. This is the state that software will want to control in order to power up and down the interface.

NORMAL_START

The execution of the powerup and powerdown sequence can be modified somewhat by choosing to use either the NORM or ALT program entry point.

SAFE_STATE

Sets the safe operating state. There are only two choices: powered up and powered down. To change this register, software should load in the new value and write SETTING_NEW=TRIGGER. Once the change has been successfully completed, the SEQ_CTL_STATUS will indicate STOPPED. The execution of the powerup and powerdown sequence can be modified somewhat by choosing to use either the standard or the alternate program entry point. This is the state that the hardware will use whenever it initiates the mode switch/shutdown procedure for this device. In general, this should be set to STATE_PD and left there always.

SAFE_START

The execution of the powerup and powerdown 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. Be sure SEQ_CTL_STATUS=STOPPED, 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 SEQ_CTL_STATUS=STOPPED. If it doesn't happen within one frame time, then Software may opt to accelerate the change by writing SOR_SEQ_CTL SWITCH=FORCE (being careful not to disturb other settings in that register).

Note: Software can start polling on SEQ_CTL_STATUS right after writing SETTING_NEW to trigger because SEQ_CTL_STATUS is set to RUNNING after 2 sorclk cycles which will always be less than the reg read delay.

Usage: boot / initialization / mode switch / normal operation / shutdown

Offset: 0x15 | Byte Offset: 0x54 | Read/Write: R/W | Reset: 0xXX000000 (0b0xxxxxxxxxxxxx00xxxxxxxxxxxxxx00)

Bit	R/W	Reset	Description
31	RW	DONE	SETTING_NEW:w1c 0 = DONE 1 = PENDING 1 = TRIGGER
28	RO	X	MODE:

Bit	R/W	Reset	Description
			0 = NORMAL 1 = SAFE
24	RO	X	HALT_DELAY: 0 = DONE 1 = ACTIVE
17	RW	NORMAL	SAFE_START: 0 = NORMAL 1 = ALT
16	RW	PD	SAFE_STATE: 0 = PD 1 = PU
1	RW	NORMAL	NORMAL_START: 0 = NORMAL 1 = ALT
0	RW	PD	NORMAL_STATE: 0 = PD 1 = PU

24.6.17 SOR_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.

HEAD_NUMBER

Report the display controller (head) that the SOR is currently attached to.

The ARM state of HEAD_NUMBER is only promoted on an attach event.

When SOR gets detached with HEAD_NUMBER set to NONE this register would still report the head that SOR was connected to before it got detached

TPAT

Selects the test generator pattern. When test generator is running, H will be high for 1 in 4 repetitions (of duration 1024 clocks), and V will be high for 4 in 16 repetitions coincident with H. Only operates when test mode is enabled.

- LO force all 0s
- TDAT uses a hardware fixed value of 1023
- RAMP test generator output ramps
- WALK test generator output is a walking one
- MAXSTEP 0, 1023, ...
- MINSTEP 511, 512, ...

CRC

The source of the data for CRC computation (for verification) can be either before the high-speed serializer logic, or post the high-speed seralizer/deserializer logic.

TESTMUX[7:0]

Test MUX select - output seen on PROBE

Usage: debug

Offset: 0x16 | Byte Offset: 0x58 | Read/Write: R/W | Reset: 0x0080XX00 (0b000000001000xx00xxxxxxxxx0xxxx0x)

Bit	R/W	Reset	Description
31:24	RW	AVSS	TESTMUX: 0 = AVSS 2 = CLOCKIN 4 = PLL_VOL 8 = SLOWCLKINT 16 = AVDD 32 = VDDREG 64 = REGREF_VDDREG 128 = REGREF_AVDD
23	RW	POST_DESERIALIZE	CRC: 0 = PRE_SERIALIZE 1 = POST_DESERIALIZE
22:20	RW	LO	TPAT: 0 = LO 1 = TDAT 2 = RAMP 3 = WALK 4 = MAXSTEP 5 = MINSTEP
17:16	RW	NORMAL	DSRC: 0 = NORMAL 1 = DEBUG 2 = TGEN
15:12	RO	X	HEAD_NUMBER: 0 = NONE 1 = HEAD0 2 = HEAD1
10	RO	X	ATTACHED: 0 = FALSE 1 = TRUE
9:8	RO	X	ACT_HEAD_OPMODE: 0 = SLEEP 1 = SNOOZE 2 = AWAKE

Bit	R/W	Reset	Description
6	RW	DISABLE	INVD: 0 = DISABLE 1 = ENABLE
1	RW	DISABLE	TEST_ENABLE: 0 = DISABLE 1 = ENABLE

24.6.18 SOR_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

VCOPD

- 1 = powerdown the VCO

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 software writes NV_PDISP_SOR_PLL0_PULLDOWN to DISABLED and sequencer pulldown is enabled, then an NV_PDISP_SOR_PLL0_PULLDOWN read will return DISABLED (while pulldown might be ENABLED due to sequencer control).

VCOCAP[3:0]

Selects the VCO capacitor and adjusts ring oscillator inter-stage load.

FILTER[3:0]

Bits 2:0 select the loop filter and adjusts the filter resistor value.

Bit 3 controls the VCO startup bit for the TMDS_DUAL macro.

- 0 - normal operation (default)
- 1 - forced VCO oscillation mode

ICHPMP[3:0]

Specifies additions to the charge pump current in steps of 0.375uA.

IOCURRENT[5:0]

Used for I/O control.

This field has been replaced by the SOR_LANE_DRIVE_CURRENT registers.

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.

- 0000 - lowest
- ...
- 1000 - default
- ...
- 1111 - highest

COMPOUT

Termination calibration status. Only exists on DisplayPort SORs.

This procedure should be performed on boot time to program TERMADJ correctly. Ideally this should be re-calibrated on Hot Plug Detect as well.

The procedure for termination calibration:

1. Start with the TERM_ADJ = 1000
2. Wait 100µs and check COMPOUT
3. If COMPOUT=1, change the MSB to 0
4. Repeat for the next significant bits until all 4 bits are determined.

SUPPLYLEV[1:0]

Supply voltage level descriptor

LOADADJ[3:0]

Load pulse position adjust

LVDS_CM[1:0]

Common mode control for LVDS

- 00 - 1.25V
- 01 - 1.30V
- 10 - 1.35V
- 11 - 1.20V

COHERENTMODE

Output reference clock selector

- 0 = Non-coherent mode
- 1 = Coherent mode

BGAP_CNTL

Band-gap current setting control.

- 00 = Normal
- 01 = Positive coefficient

- 10 = Negative coefficient
- 11 = Normal + 12.5%

LOCKDET

Status signal indicating whether PLL is locked within the desired resolution.

- 0 = Not locked
- 1 = Locked

MISC_CNTL

Reserved.

DCIR_PLL_RESET

For the Display-over-PCIe feature, a different type of analog macro was used in the SOR.

The PLL in this macro needs to be held in reset during mode switches when the reference clock drops to a safe clock frequency. A signal from the DCIR asserts this reset. NV_PDISP_SOR_PLL2_DCIR_PLL_RESET can be used to override this signal

- OVERRIDE: The signal from the DCIR cannot power down the PLL.
- ALLOW: The signal from the DCIR will be able to power down the PLL.

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 gate seq_2all_pll_pulldown from PULLDOWN. this can be used to remove sequencer control to PULLDOWN.

- SEQ_PLL_PULLDOWN_OVERRIDE: Mask the pll_pulldown signal from the SOR sequencer so that it has no effect on the PLL_PULLDOWN port.
- SEQ_PLL_PULLDOWN_ALLOW: Allow the pll_pulldown signal from the SOR sequencer to affect the PLL_PULLDOWN port.

AUX1 Power on override for PLLCAPPD. The PLL in the analog macro will not run unless this is powered on. This can be used if the PLL needs to be powered on before the sequencer runs.

- SEQ_PLLCAPPD_OVERRIDE: Mask the pll_reset signal from the SOR sequencer so that it has no effect on the PLLCPAPPD port of the SOR macro.
- SEQ_PLLCAPPD_ALLOW: Allow the pll_reset signal from the SOR sequencer to assert PLLCAPPD.

AUX2 gate PDBG from sequencer_pd_macro.

This can be used to force the Bandgap to power on.

- OVERRIDE_POWERDOWN: gate the PDBG signal from the sequencer, PDBG can still be set by AUX6
- ALLOW_POWERDOWN:normal operation, the PDBG signal from the sequencer can power down the bandgap

AUX3 rotate grn channel by 1 bit to reduce tmds EMI

- ROTATE_DISABLE: Do not rotate
- ROTATE_ENABLE: Right-rotate green channel by 1 bit

AUX4 has been used to enable duplicate controls for the Dual link HDCP. This is needed for some Dual Link panels.

- DUPLICATE_CTRL_ENABLE: Copy the control bits from the primary link to the secondary.
- DUPLICATE_CTRL_DISABLE: Zero out the ctrl bits on the secondary link.

AUX5 should be set to DUAL_LINK_LVDS for dual link lvds. this field it used to distinguish between single link and dual link LVDS

- DUAL_LINK_LVDS: Used for dual link LVDS.
- SINGLE_LINK_LVDS: Used for single link LVDS.

AUX6 Main power down for bandgap and reference current setting circuitry. (NOTE: This must be BANDGAP_POWERDOWN_DISABLE when in power down mode if 3.3V is seen on the IO_[A/B]_VDD rails)

- BANDGAP_POWERDOWN_DISABLE: Allow the bandgap to power up.
- BANDGAP_POWERDOWN_ENABLE: Force the bandgap to power down.

AUX7 Controls PDPORT in the analog macro which powers down all the output links/lanes. For instance, txd{n/p}0-9 for dual TMDS macro.

- PORT_POWERDOWN_DISABLE: Allow the output ports to be turned on.
- PORT_POWERDOWN_ENABLE: Force the output ports to power down. This will prevent any data from reaching the sink device.

AUX8 Power on enforcement for PLLCAPPD. This can force the PLLCAPPD to be asserted to reset the analog macro.

This can be used to force the PLLCAPPD to be asserted for work arounds where it is inconvenient to run the SOR sequencer.

- SEQ_PLLCAPPD_ENFORCE_ENABLE: Force PLLCAPPD to be asserted.
- SEQ_PLLCAPPD_ENFORCE_DISABLE: Not to force PLLCAPPD, normal state.

AUX9 Overrides LVDSEN programmed by priv register. This can force the LVDSEN to be deasserted.

- LVDSEN_ALLOW: Allow priv register to assert LVDSEN.
- LVDSEN_OVERRIDE: Override LVDSEN to be deasserted, mask the priv register setting for LVDSEN.

Usage: boot / initialization / mode switch

Offset: 0x17 | Byte Offset: 0x5c | Read/Write: R/W | Reset: 0x0f0003d5 (0bxxxx1111xxxx0000xx0000111101x1x1)

Bit	Reset	Description
27:24	RST	ICHPMP: 15 = RST
19:16	RST	FILTER: 0 = RST
13:12	V25	TXREG_LEVEL: 0 = V25 1 = V15 2 = V35 3 = V45
11:8	RST	VCOCAP: 3 = RST
7:6	V45	PLLREG_LEVEL: 0 = V25 1 = V15 2 = V35 3 = V45
5	DISABLE	PULLDOWN: 0 = DISABLE 1 = ENABLE
4	EXT	RESISTORSEL: 0 = INT 1 = EXT 1 = RST

Bit	Reset	Description
2	ASSERT	VCOPD: 0 = RESCIND 1 = ASSERT
0	OFF	PWR: 0 = ON 1 = OFF

24.6.19 SOR_NV_PDISP_SOR_PLL1_0

Usage: boot / initialization / mode switch

Offset: 0x18 | Byte Offset: 0x60 | Read/Write: R/W | Reset: 0x0000X000 (0bxx0xxx000000xxxxxxx10000xx000000)

Bit	R/W	Reset	Description
29	RW	DISABLE	COHERENTMODE: 0 = DISABLE 1 = ENABLE
25:24	RW	0x0	LVDSCM
23:20	RW	CENTER	LOADADJ: 0 = CENTER
15	RO	X	TERM_COMPOUT: 0 = LOW 1 = HIGH
12:9	RW	OHM500	TMDS_TERMADJ: 8 = OHM500
8	RW	DISABLE	TMDS_TERM: 0 = DISABLE 1 = ENABLE
5:0	RW	RST	IOCURRENT: 0 = RST

24.6.20 SOR_NV_PDISP_SOR_PLL2_0

Usage: boot / initialization / mode switch

Offset: 0x19 | Byte Offset: 0x64 | Read/Write: R/W | Reset: 0x01c00000 (0bxxxxxx01110000000000000000xx00)

Bit	Reset	Description
25	LVDSEN_ALLOW	AUX9: 0 = LVDSEN_ALLOW 1 = LVDSEN_OVERRIDE
24	SEQ_PLLCAPPD_ENFORCE_ENABLE	AUX8: 0 = SEQ_PLLCAPPD_ENFORCE_DISABLE 1 = SEQ_PLLCAPPD_ENFORCE_ENABLE
23	PORT_POWERDOWN_ENABLE	AUX7: 0 = PORT_POWERDOWN_DISABLE 1 = PORT_POWERDOWN_ENABLE
22	BANDGAP_POWERDOWN_ENABLE	AUX6: 0 = BANDGAP_POWERDOWN_DISABLE 1 = BANDGAP_POWERDOWN_ENABLE
21	SINGLE_LINK_LVDS	AUX5: 0 = SINGLE_LINK_LVDS 1 = DUAL_LINK_LVDS
20	DUPLICATE_CTRL_DISABLE	AUX4: 0 = DUPLICATE_CTRL_DISABLE

Bit	Reset	Description
		1 = DUPLICATE_CTRL_ENABLE
19	ROTATE_DISABLE	AUX3: 0 = ROTATE_DISABLE 1 = ROTATE_ENABLE
18	OVERRIDE_POWERDOWN	AUX2: 0 = OVERRIDE_POWERDOWN 1 = ALLOW_POWERDOWN
17	SEQ_PLLCAPPD_ALLOW	AUX1: 0 = SEQ_PLLCAPPD_ALLOW 1 = SEQ_PLLCAPPD_OVERRIDE
16	SEQ_PLL_PULLDOWN_ALLOW	AUX0: 0 = SEQ_PLL_PULLDOWN_ALLOW 1 = SEQ_PLL_PULLDOWN_OVERRIDE
15:12	0x0	PLL_MDIV
11:8	BY_3	PLL_NDIV: 0 = BY_3 1 = BY_5 2 = BY_6 3 = BY_7 4 = BY_10
7:4	BY_1	PLL_PDIV: 0 = BY_1 1 = BY_2 2 = BY_2B 3 = BY_4
1	DISABLE	DIV_RATIO_OVERRIDE: 0 = DISABLE 1 = ENABLE
0	OVERRIDE	DCIR_PLL_RESET: 0 = OVERRIDE 1 = ALLOW

24.6.21 SOR_NV_PDISP_SOR_PLL3_0

SOR_PLL3

This register directly controls the SOR analog macro.

KICKSTART

- _DISABLE: Disable kickstart
- _LOOP_40: short loop-filter 40% of avdd14
- _LOOP_60: short loop-filter 60% of avdd14
- _LOOP_50: short loop-filter 50% of avdd14

AVDD14_LEVEL

Internal regulated 1.4V voltage level control bits

- _1_35V: 1.35V
- _1_40V: 1.40V
- _1_45V: 1.45V
- _1_50V: 1.50V

No other values are valid for this field.

AVDD10_LEVEL

Internal regulated 1.0V voltage level control bits

- _0_95V: 0.95V
- _1_00V: 1.00V
- _1_05V: 1.05V
- _1_10V: 1.10V

No other values are valid for this field.

CLKDIST_MODE

Determines the clock distribution by CMOS buffer or CML buffers:

- _CMOS: CMOS buffers
- _CML: CML buffers

PLLVDD_MODE

Field to determine the PLL voltage. Software needs to program this after power-on.

- _1_8V: 1.8V --> For LVDS
- _3_3V: 3.3V --> For eDP

PLL_BYPASS

Enable the PLL bypass mode, allowing the reference clock to drive the output driver directly to all channels. This would be used for debug purposes only.

- _ENABLE: Force the PLL output onto all lanes.
- _DISABLE: Normal operation

TEST_REFCLK_EN

Enable the TEST_REFCLK to drive the output driver directly, for each individual lane

This is a bit mask with one bit per lane. This field is for debug purposes only and Software should not be using this field.

BG_VREF_LEVEL

Control bits for the bandgap's output voltages

V(bandgap output) = 0.7V * (84% + BG_VREF_LEVEL*2%)

i.e.

- 0000 -> 0.588V
- ...
- 1000 -> 0.7V
- ...
- 1111 -> 0.798V

BG_TEMP_COEF

Control bits for the bandgap's temperature coefficient.

Offset: 0x1a | Byte Offset: 0x68 | Read/Write: R/W | Reset: 0x38002220 (0b00111000000000000x01000100010xx00)

Bit	Reset	Description
31:28	0x3	BG_TEMP_COEF
27:24	0x8	BG_VREF_LEVEL
23:16	0x0	TEST_REFCLK_EN
14	DISABLE	PLL_BYPASS: 0 = DISABLE 1 = ENABLE
13	V3_3	PLLVDD_MODE: 0 = V1_8 1 = V3_3
12	CMOS	CLKDIST_MODE: 0 = CMOS 1 = CML
11:8	V1_05	AVDD10_LEVEL: 0 = V0_95 1 = V1_00 2 = V1_05 3 = V1_10
7:4	V1_45	AVDD14_LEVEL: 0 = V1_35 1 = V1_40 2 = V1_45 3 = V1_50
1:0	DISABLE	KICKSTART: 0 = DISABLE 1 = LOOP_40 2 = LOOP_50 3 = LOOP_60

24.6.22 SOR_NV_PDISP_SOR_CSTM_0

The class permits the driver to select a number of operating modes for the SOR. Some of these modes (TMDS modes) are well defined and stable. Some modes may require customization by software before they will work. These registers are used for that customization. The fields available for customization are:

PD_TXDA[3:0]

Bitwise control to power down the data pins of link A. Set to DISABLE to power down the pin.

PD_TXDB[3:0]

Bitwise control to power down the data pins of link B. Set to DISABLE to power down the pin.

PD_TXCA

Power down the clock pin of link A. Set to DISABLE to power down the pin.

PD_TXCB

Power down the clock pin of link B. Set to DISABLE to power down the pin.

UPPER

Designates whether LVDS bank A is the upper, odd, or first pixel. Bank B is always set to !UPPER. The serial output from UPPER=1 will be Clock and A0-A3. The serial output from UPPER=0 (and DUALMODE) will be Clock and A4-A7. 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

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.

For all the TMDS modes, ROTCLK field is programmable using NV_PDISP_SOR_CSTM_ROTCLK.

ROTDAT[2:0]

Before encoding the 8 bits of each color channel, the 8 bits within each same amount. ROTDAT should be between 0 and 7. For example, to support 24.1 LVDS data format, set ROTDAT = 6, then input channel data (r7,r6,r5,r4,r3,r2,r1,r0} would become {r5,r4,r3,r2,r1,r1,r7,r6}.

TMDS modes of operation are standard and stable. The table below summarizes the fixed values the hardware uses for the above fields for TMDS operating modes.

```
                                                DUAL
                SINGLE      SINGLE      SINGLE  SINGLE      DUAL
                TMDS_A      TMDS_B      TMDS_AB TMDS        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]   |  X           X           X       X           X   |
ROTDAT[2:0]   |  0           0           0       0           0   |
                ------------------------------------------------
Where X (ROTCLK[3:0]) = NV_PDISP_SOR_CSTM_ROTCLK;
```

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. Software 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: 0x1b | Byte Offset: 0x6c | Read/Write: R/W | Reset: 0x0001c800 (0bx0000000xx0x000111001x0000000000)

Bit	Reset	Description
30:28	RST	ROTDAT: 0 = RST
27:24	RST	ROTCLK: 0 = RST
21	BY_7	PLLDIV: 0 = BY_7 1 = BY_10
19	DISABLE	BALANCED: 0 = DISABLE 1 = ENABLE
18	DISABLE	NEW_MODE: 0 = DISABLE 1 = ENABLE
17	DISABLE	DUP_SYNC: 0 = DISABLE 1 = ENABLE
16	ENABLE	LVDS_EN: 0 = DISABLE 1 = ENABLE
15	ENABLE	LINKACTB: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
14	ENABLE	LINKACTA: 0 = DISABLE 1 = ENABLE
13:12	LVDS	MODE: 0 = LVDS 1 = DP
11	TRUE	UPPER: 0 = FALSE 1 = TRUE
9	ENABLE	PD_TXCB: 0 = ENABLE 1 = DISABLE
8	ENABLE	PD_TXCA: 0 = ENABLE 1 = DISABLE
7	ENABLE	PD_TXDB_3: 0 = ENABLE 1 = DISABLE
6	ENABLE	PD_TXDB_2: 0 = ENABLE 1 = DISABLE
5	ENABLE	PD_TXDB_1: 0 = ENABLE 1 = DISABLE
4	ENABLE	PD_TXDB_0: 0 = ENABLE 1 = DISABLE
3	ENABLE	PD_TXDA_3: 0 = ENABLE 1 = DISABLE
2	ENABLE	PD_TXDA_2: 0 = ENABLE 1 = DISABLE
1	ENABLE	PD_TXDA_1: 0 = ENABLE 1 = DISABLE
0	ENABLE	PD_TXDA_0: 0 = ENABLE 1 = DISABLE

24.6.23 SOR_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: 0x1c | Byte Offset: 0x70 | Read/Write: R/W | Reset: 0x00XXXX0X (0bx0000000xxxx000x1xxxxx0x00000xxx)

Bit	R/W	Reset	Description
30:28	RW	RST	ROTDAT: 0 = RST
27:24	RW	RST	ROTCLK: 0 = RST
21	RO	X	PLLDIV:

Bit	R/W	Reset	Description
			0 = BY_7
19	RW	DISABLE	BALANCED: 0 = DISABLE 1 = ENABLE
18	RW	DISABLE	NEW_MODE: 0 = DISABLE 1 = ENABLE
17	RW	DISABLE	DUP_SYNC: 0 = DISABLE 1 = ENABLE
16	RO	X	LVDS_EN: 1 = ENABLE
15	RW	ENABLE	LINKACTB: 0 = DISABLE 1 = ENABLE
14	RO	X	LINKACTA: 1 = ENABLE
13:12	RO	X	MODE: 0 = LVDS
11	RO	X	UPPER: 0 = FALSE 1 = TRUE
9	RW	ENABLE	PD_TXCB: 0 = ENABLE 1 = DISABLE
8	RO	X	PD_TXCA: 0 = ENABLE
7	RW	ENABLE	PD_TXDB_3: 0 = ENABLE 1 = DISABLE
6	RW	ENABLE	PD_TXDB_2: 0 = ENABLE 1 = DISABLE
5	RW	ENABLE	PD_TXDB_1: 0 = ENABLE 1 = DISABLE
4	RW	ENABLE	PD_TXDB_0: 0 = ENABLE 1 = DISABLE
3	RW	ENABLE	PD_TXDA_3: 0 = ENABLE 1 = DISABLE
2	RO	X	PD_TXDA_2: 0 = ENABLE
1	RO	X	PD_TXDA_1: 0 = ENABLE
0	RO	X	PD_TXDA_0: 0 = ENABLE

24.6.24 SOR_NV_PDISP_SOR_CRCA_0

The following registers are used to fetch CRCs when running VGA mode tests. Three registers are used. One contains the valid bit, one contains the actual computed CRC, and the third contains the error (overrun bit). Proper use of these registers is as follows:

1) Poll SOR_CRCA for VALID == TRUE.

2) Reset SOR_CRCA by writing RESET to it.

3) Read SOR_CRCB to get the CRC

Usage: verif

Offset: 0x1d | Byte Offset: 0x74 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	VALID:w1c(RST) 0 = FALSE 1 = TRUE 1 = RST

24.6.25 SOR_NV_PDISP_SOR_CRCB_0

Usage: verif

Offset: 0x1e | Byte Offset: 0x78 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CRC:w1c

24.6.26 SOR_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: 0x1f | Byte Offset: 0x7c | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	R/W	Reset	Description
2	RO	X	STATUS: 0 = NOT_BLANKED 1 = BLANKED
1	RW	IMMEDIATE	TRANSITION: 0 = IMMEDIATE

Bit	R/W	Reset	Description
			1 = NEXT_VSYNC
0	RW	FALSE	OVERRIDE: 0 = FALSE 1 = TRUE

24.6.27 SOR_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 trisate 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 powerup program sequence. Always 0.

PU_PC_ALT

The alternate entry point into the powerup program sequence. Defaults to the same value as PU_PC.

PD_PC

The program counter for the start of the powerdown program sequence. Defaults to 8, evenly dividing the available program space.

PD_PC_ALT

The alternate entry point into the powerdown 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.

Usage: boot / initialization / mode switch

Offset: 0x20 | Byte Offset: 0x80 | Read/Write: R/W | Reset: 0xX00X880X (0bxxxxxxxxxxxxxxxx100010000000xxxx)

Bit	R/W	Reset	Description
30	RW	X	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

Bit	R/W	Reset	Description
3:0	RO	X	PU_PC

24.6.28 SOR_NV_PDISP_SOR_LANE_SEQ_CTL_0

The SOR_LANE_SEQ_CTL register is used to control the operation of the SOR lane sequencer. The lane sequencer is used to power up or power down the lanes in the analog macro one at a time to prevent spikes on the power rail.

LANE_STATE

This is the current status of each lane's power.

```
        |    DP          TMDS            LVDS
 -------+---------------------------------------
  LANE0 | DPA lane2   LinkA Lane0     LinkA Lane0
  LANE1 | DPA lane1   LinkA Lane1     LinkA Lane1
  LANE2 | DPA lane0   LinkA Lane2     LinkA Lane2
  LANE3 | unused      unused          LinkA Lane3
  LANE4 | DPA lane3   LinkA clock     LinkA clock
  LANE5 | DPB lane2   LinkB Lane0     LinkB Lane0
  LANE6 | DPB lane1   LinkB Lane1     LinkB Lane1
  LANE7 | DPB lane0   LinkB lane2     LinkB Lane2
  LANE8 | unused      unused          LinkB Lane3
  LANE9 | DPB lane3   LinkB clock     LinkB clock
```

DELAY

Number of microseconds to delay between each lanes' power state change. The recommended value is 1 µs.

NEW_POWER_STATE

This controls whether the lanes should be powered up or powered down when this sequencer is triggered by SETTING_NEW_TRIGGER.

- PU: Power up the requested lanes
- PD: Power down to the requested lanes

SEQUENCE

Controls the direction of the power up or power down sequence.

- UP: Lanes are powered up in increasing order
- DOWN: Lanes are powered up in decreasing order

STATE

Whenever this sequencer is running, this field will be set to _BUSY.

SETTING_NEW

This bit can be used to run this sequencer outside of the normal SOR sequencer operation. If the sequencer is already BUSY because of a request from the SOR sequencer, the lane sequencer will wait until that request is complete before servicing this request. It is recommended that this trigger should only be used when the SOR sequencer is not running in order to prevent this type of interaction.

The lane sequencer looks at the current protocol, NV_PDISP_SOR_DP_LINKCTL_ENABLE and whether the request is power-up or power-down to determine what the new lane state should be.

```
DP_LINKCTL_ENABLE   PowerUp   Protocol==DP || New Lane State
-----------------------------------------++----------------
        0             0           X       ||   All lanes
        0             1           0       || Based on Protocol
        0             1           1       || DP_PADCTL_PD_TXD*
        1             X           X       || DP_PADCTL_PD_TXD*
```

Usage: boot / initialization / mode switch

Offset: 0x21 | Byte Offset: 0x84 | Read/Write: R/W | Reset: 0xX0011XXX (0b0xxxxxxxxxx0xxx10001xxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	DONE	SETTING_NEW: 0 = DONE 1 = PENDING 1 = TRIGGER
28	RO	X	SEQ_STATE: 0 = IDLE 1 = BUSY
20	RW	UP	SEQUENCE: 0 = UP 1 = DOWN
16	RW	PD	NEW_POWER_STATE: 0 = PU 1 = PD
15:12	RW	0x1	DELAY
9	RO	X	LANE9_STATE: 0 = POWERUP 1 = POWERDOWN
8	RO	X	LANE8_STATE: 0 = POWERUP 1 = POWERDOWN
7	RO	X	LANE7_STATE: 0 = POWERUP 1 = POWERDOWN
6	RO	X	LANE6_STATE: 0 = POWERUP 1 = POWERDOWN
5	RO	X	LANE5_STATE: 0 = POWERUP 1 = POWERDOWN
4	RO	X	LANE4_STATE: 0 = POWERUP 1 = POWERDOWN
3	RO	X	LANE3_STATE: 0 = POWERUP 1 = POWERDOWN
2	RO	X	LANE2_STATE: 0 = POWERUP 1 = POWERDOWN
1	RO	X	LANE1_STATE: 0 = POWERUP 1 = POWERDOWN

Bit	R/W	Reset	Description
0	RO	X	LANE0_STATE: 0 = POWERUP 1 = POWERDOWN

24.6.29 SOR_NV_PDISP_SOR_SEQ_INST0_0

This register set is used to preload the power-up and power-down sequences. Each step of the program performs the following based on the fields below. First, either delay for WAIT time or until the next leading edge of vertical sync from the RG. The range of WAIT is 0-1023 us or 0-1023 ms. Second, set sequencer outputs (A, B, I/O pin powerdown override etc) to the values specified in the instruction. The meaning and use of pins A and B is chip and board dependent i.e. something the SW will need to know how to configure. Lastly, if HALT==1, stop, otherwise, execute the next instruction.

WAIT_TIME

Amount of time to wait (in either microseconds or milliseconds). Note if WAIT_UNITS=VSYNC, the WAIT_TIME field is ignored.

WAIT_UNITS

Wait delay mode, µs, ms, vsync.

PDPLL

Asserts the PDPLL port on the analog macro, powering down the PLL. This sequencer control can be overridden by NV_PDISP_SOR_PLL2_AUX1_SEQ_PLLCAPPD_OVERRIDE. NV_PDISP_SOR_PLL0_PWR_OFF will force PDPLL to be asserted, regardless of the state of the sequencer.

- YES: Power down the PLL.
- NO: Do not power down the PLL

PDPORT

Asserts the PDPORT port on the SOR analog macro. This powers down all output lanes. This sequencer control can be overridden by NV_PDISP_SOR_DP_LINKCTL_ENABLE. NV_PDISP_SOR_PLL2_AUX7_PORT_POWERDOWN_ENABLE will force PDPORT to be asserted, regardless of the state of the sequencer.

- YES: Power down the port
- NO: Do not power down the port

LANE_SEQ

When set to _RUN, the sequencer will kick off the Lane Sequencer (see NV_PDISP_SOR_LANE_SEQ_CTL). The SOR sequencer will not continue to the next instruction until the lane sequencer completes. The lane sequencer will run before the other commands in the current instruction take effect and before the delay in the current instruction.

When running the Power Up sequence, the lanes to be powered up are determined by the new active protocol. When running the Power Down sequence, all lanes are powered down.

```
DP_LINKCTL_ENABLE   PowerUp   Protocol==DP || New Lane State
-----------------------------------------++---------------
        0             0           X      ||   All lanes
        0             1           0      || Based on Protocol
        0             1           1      || DP_PADCTL_PD_TXD
        1             X           X      || DP_PADCTL_PD_TXD
```

SEQUENCE

This field controls the direction of the lane power up or power down sequence.

- UP: Lanes are powered up or down in increasing order PDTXD_0, PDTXD_1 .. PDTXD_9
- DOWN: Lanes are powered up or down in decreasing order PDTXD_9, PDTXD_8 .. PDTXD_0

PIN_A, PIN_B

State to drive the external pin to. The connection state of these pins is controlled in either the host or the PCB manager.

DRIVE_PWM_OUT_LO

Drive the PWM circuit output low. Note that this driven low condition is applied before the optional PWM_POLARITY control, so that the external pin can actually be driven high or low.

TRISTATE_IOS

Force all output pins to tristate.

- 0 = enable output pins
- 1 = disable/tristate output pins

For chips with the SOR Lane Sequencer, this does not take effect until the Lane Sequencer is run. Since Lane Sequencer runs before the values in the current instruction take effect, TRISTATE_IOS should be set to the desired value on the instruction before LANE_SEQ_RUN.

BLACK_DATA

Override the pixel bus from the RG, and replace the pixels with black.

- 0 = normal data output
- 1 = force black output

BLANK_DE, BLANK_H, BLANK_V

Override the de, hsync, and vsync signals from the rg. The signals will be forced to the inactive, i.e., deasserted state.

- 0 = normal output
- 1 = force signal inactive/deasserted

ASSERT_PLL_RESET

Assert reset to the PLL (PLLCAPPD). This sequencer control can be overridden by NV_PDISP_SOR_PLL2_AUX1_SEQ_PLLCAPPD_OVERRIDE.
NV_PDISP_SOR_PLL2_AUX8_SEQ_PLLCAPPD_ENFORCE_ENABLE will force PLLCAPPD to be asserted, regardless of the state of the sequencer.

- 0 = normal
- 1 = force reset

POWERDOWN_MACRO

Override LINKACTA, LINKACTB and force LVDS/TMDS macro to powerdown state (Only controls PDBG)

- 0 = normal operation
- 1 = force powerdown

PULLDOWN

Weak pulldown enable (Only exists on TMDS/LVDS SORs)

- 1 = 2Kohm pulldown on all outputs.
- 0 = Weak pulldown disabled.

Offset: 0x22 | Byte Offset: 0x88 | Read/Write: R/W | Reset: 0x01008000 (0b00000001000x00001x00xx0000000000)

Bit	Reset	Description
31	DISABLE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	NORMAL	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	NORMAL	ASSERT_PLL_RESET: 0 = NORMAL 1 = RST
28	NORMAL	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	NORMAL	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	NORMAL	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	NORMAL	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	TRISTATE	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	FALSE	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	LOW	PIN_B: 0 = LOW 1 = HIGH
21	LOW	PIN_A: 0 = LOW 1 = HIGH
19	UP	SEQUENCE: 0 = UP 1 = DOWN
18	STOP	LANE_SEQ: 0 = STOP 1 = RUN
17	NO	PDPORT: 0 = NO 1 = YES
16	NO	PDPLL: 0 = NO 1 = YES
15	TRUE	HALT: 0 = FALSE 1 = TRUE
13:12	US	WAIT_UNITS: 0 = US

Bit	Reset	Description
		1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

24.6.30 SOR_NV_PDISP_SOR_SEQ_INST1_0

Offset: 0x23 | Byte Offset: 0x8c | Read/Write: R/W | Reset: 0x01008000 (0b00000001000x00001x00xx0000000000)

Bit	Reset	Description
31	DISABLE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	NORMAL	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	NORMAL	ASSERT_PLL_RESET: 0 = NORMAL 1 = RST
28	NORMAL	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	NORMAL	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	NORMAL	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	NORMAL	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	TRISTATE	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	FALSE	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	LOW	PIN_B: 0 = LOW 1 = HIGH
21	LOW	PIN_A: 0 = LOW 1 = HIGH
19	UP	SEQUENCE: 0 = UP 1 = DOWN
18	STOP	LANE_SEQ: 0 = STOP 1 = RUN
17	NO	PDPORT: 0 = NO 1 = YES
16	NO	PDPLL: 0 = NO 1 = YES

Bit	Reset	Description
15	TRUE	HALT: 0 = FALSE 1 = TRUE
13:12	US	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

24.6.31 SOR_NV_PDISP_SOR_SEQ_INST2_0

Offset: 0x24 | Byte Offset: 0x90 | Read/Write: R/W | Reset: 0x01008000 (0b00000001000x00001x00xx0000000000)

Bit	Reset	Description
31	DISABLE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	NORMAL	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	NORMAL	ASSERT_PLL_RESET: 0 = NORMAL 1 = RST
28	NORMAL	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	NORMAL	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	NORMAL	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	NORMAL	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	TRISTATE	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	FALSE	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	LOW	PIN_B: 0 = LOW 1 = HIGH
21	LOW	PIN_A: 0 = LOW 1 = HIGH
19	UP	SEQUENCE: 0 = UP 1 = DOWN
18	STOP	LANE_SEQ: 0 = STOP 1 = RUN
17	NO	PDPORT: 0 = NO

Bit	Reset	Description
		1 = YES
16	NO	PDPLL: 0 = NO 1 = YES
15	TRUE	HALT: 0 = FALSE 1 = TRUE
13:12	US	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

24.6.32 SOR_NV_PDISP_SOR_SEQ_INST3_0

Offset: 0x25 | Byte Offset: 0x94 | Read/Write: R/W | Reset: 0x01008000 (0b00000001000x00001x00xx0000000000)

Bit	Reset	Description
31	DISABLE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	NORMAL	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	NORMAL	ASSERT_PLL_RESET: 0 = NORMAL 1 = RST
28	NORMAL	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	NORMAL	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	NORMAL	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	NORMAL	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	TRISTATE	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	FALSE	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	LOW	PIN_B: 0 = LOW 1 = HIGH
21	LOW	PIN_A: 0 = LOW 1 = HIGH
19	UP	SEQUENCE: 0 = UP 1 = DOWN

Bit	Reset	Description
18	STOP	LANE_SEQ: 0 = STOP 1 = RUN
17	NO	PDPORT: 0 = NO 1 = YES
16	NO	PDPLL: 0 = NO 1 = YES
15	TRUE	HALT: 0 = FALSE 1 = TRUE
13:12	US	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

24.6.33 SOR_NV_PDISP_SOR_SEQ_INST4_0

Offset: 0x26 | Byte Offset: 0x98 | Read/Write: R/W | Reset: 0x01008000 (0b00000001000x00001x00xx0000000000)

Bit	Reset	Description
31	DISABLE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	NORMAL	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	NORMAL	ASSERT_PLL_RESET: 0 = NORMAL 1 = RST
28	NORMAL	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	NORMAL	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	NORMAL	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	NORMAL	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	TRISTATE	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	FALSE	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	LOW	PIN_B: 0 = LOW 1 = HIGH
21	LOW	PIN_A: 0 = LOW

Bit	Reset	Description
		1 = HIGH
19	UP	SEQUENCE: 0 = UP 1 = DOWN
18	STOP	LANE_SEQ: 0 = STOP 1 = RUN
17	NO	PDPORT: 0 = NO 1 = YES
16	NO	PDPLL: 0 = NO 1 = YES
15	TRUE	HALT: 0 = FALSE 1 = TRUE
13:12	US	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

24.6.34 SOR_NV_PDISP_SOR_SEQ_INST5_0

Offset: 0x27 | Byte Offset: 0x9c | Read/Write: R/W | Reset: 0x01008000 (0b00000001000x00001x00xx0000000000)

Bit	Reset	Description
31	DISABLE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	NORMAL	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	NORMAL	ASSERT_PLL_RESET: 0 = NORMAL 1 = RST
28	NORMAL	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	NORMAL	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	NORMAL	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	NORMAL	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	TRISTATE	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	FALSE	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE

Bit	Reset	Description
22	LOW	PIN_B: 0 = LOW 1 = HIGH
21	LOW	PIN_A: 0 = LOW 1 = HIGH
19	UP	SEQUENCE: 0 = UP 1 = DOWN
18	STOP	LANE_SEQ: 0 = STOP 1 = RUN
17	NO	PDPORT: 0 = NO 1 = YES
16	NO	PDPLL: 0 = NO 1 = YES
15	TRUE	HALT: 0 = FALSE 1 = TRUE
13:12	US	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

24.6.35 SOR_NV_PDISP_SOR_SEQ_INST6_0

Offset: 0x28 | Byte Offset: 0xa0 | Read/Write: R/W | Reset: 0x01008000 (0b00000001000x00001x00xx0000000000)

Bit	Reset	Description
31	DISABLE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	NORMAL	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	NORMAL	ASSERT_PLL_RESET: 0 = NORMAL 1 = RST
28	NORMAL	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	NORMAL	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	NORMAL	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	NORMAL	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	TRISTATE	TRISTATE_IOS: 0 = ENABLE_PINS

Bit	Reset	Description
		1 = TRISTATE
23	FALSE	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	LOW	PIN_B: 0 = LOW 1 = HIGH
21	LOW	PIN_A: 0 = LOW 1 = HIGH
19	UP	SEQUENCE: 0 = UP 1 = DOWN
18	STOP	LANE_SEQ: 0 = STOP 1 = RUN
17	NO	PDPORT: 0 = NO 1 = YES
16	NO	PDPLL: 0 = NO 1 = YES
15	TRUE	HALT: 0 = FALSE 1 = TRUE
13:12	US	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

24.6.36 SOR_NV_PDISP_SOR_SEQ_INST7_0

Offset: 0x29 | Byte Offset: 0xa4 | Read/Write: R/W | Reset: 0x01008000 (0b00000001000x00001x00xx0000000000)

Bit	Reset	Description
31	DISABLE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	NORMAL	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	NORMAL	ASSERT_PLL_RESET: 0 = NORMAL 1 = RST
28	NORMAL	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	NORMAL	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	NORMAL	BLANK_DE: 0 = NORMAL 1 = INACTIVE

Bit	Reset	Description
25	NORMAL	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	TRISTATE	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	FALSE	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	LOW	PIN_B: 0 = LOW 1 = HIGH
21	LOW	PIN_A: 0 = LOW 1 = HIGH
19	UP	SEQUENCE: 0 = UP 1 = DOWN
18	STOP	LANE_SEQ: 0 = STOP 1 = RUN
17	NO	PDPORT: 0 = NO 1 = YES
16	NO	PDPLL: 0 = NO 1 = YES
15	TRUE	HALT: 0 = FALSE 1 = TRUE
13:12	US	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

24.6.37 SOR_NV_PDISP_SOR_SEQ_INST8_0

Offset: 0x2a | Byte Offset: 0xa8 | Read/Write: R/W | Reset: 0x01008000 (0b00000001000x00001x00xx0000000000)

Bit	Reset	Description
31	DISABLE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	NORMAL	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	NORMAL	ASSERT_PLL_RESET: 0 = NORMAL 1 = RST
28	NORMAL	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	NORMAL	BLANK_H: 0 = NORMAL

Bit	Reset	Description
		1 = INACTIVE
26	NORMAL	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	NORMAL	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	TRISTATE	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	FALSE	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	LOW	PIN_B: 0 = LOW 1 = HIGH
21	LOW	PIN_A: 0 = LOW 1 = HIGH
19	UP	SEQUENCE: 0 = UP 1 = DOWN
18	STOP	LANE_SEQ: 0 = STOP 1 = RUN
17	NO	PDPORT: 0 = NO 1 = YES
16	NO	PDPLL: 0 = NO 1 = YES
15	TRUE	HALT: 0 = FALSE 1 = TRUE
13:12	US	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

24.6.38 SOR_NV_PDISP_SOR_SEQ_INST9_0

Offset: 0x2b | Byte Offset: 0xac | Read/Write: R/W | Reset: 0x01008000 (0b00000001000x00001x00xx0000000000)

Bit	Reset	Description
31	DISABLE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	NORMAL	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	NORMAL	ASSERT_PLL_RESET: 0 = NORMAL 1 = RST

Bit	Reset	Description
28	NORMAL	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	NORMAL	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	NORMAL	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	NORMAL	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	TRISTATE	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	FALSE	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	LOW	PIN_B: 0 = LOW 1 = HIGH
21	LOW	PIN_A: 0 = LOW 1 = HIGH
19	UP	SEQUENCE: 0 = UP 1 = DOWN
18	STOP	LANE_SEQ: 0 = STOP 1 = RUN
17	NO	PDPORT: 0 = NO 1 = YES
16	NO	PDPLL: 0 = NO 1 = YES
15	TRUE	HALT: 0 = FALSE 1 = TRUE
13:12	US	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

24.6.39 SOR_NV_PDISP_SOR_SEQ_INSTA_0

Offset: 0x2c | Byte Offset: 0xb0 | Read/Write: R/W | Reset: 0x01008000 (0b00000001000x00001x00xx0000000000)

Bit	Reset	Description
31	DISABLE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	NORMAL	POWERDOWN_MACRO: 0 = NORMAL

Bit	Reset	Description
		1 = POWERDOWN
29	NORMAL	ASSERT_PLL_RESET: 0 = NORMAL 1 = RST
28	NORMAL	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	NORMAL	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	NORMAL	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	NORMAL	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	TRISTATE	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	FALSE	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	LOW	PIN_B: 0 = LOW 1 = HIGH
21	LOW	PIN_A: 0 = LOW 1 = HIGH
19	UP	SEQUENCE: 0 = UP 1 = DOWN
18	STOP	LANE_SEQ: 0 = STOP 1 = RUN
17	NO	PDPORT: 0 = NO 1 = YES
16	NO	PDPLL: 0 = NO 1 = YES
15	TRUE	HALT: 0 = FALSE 1 = TRUE
13:12	US	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

24.6.40 SOR_NV_PDISP_SOR_SEQ_INSTB_0

Offset: 0x2d | Byte Offset: 0xb4 | Read/Write: R/W | Reset: 0x01008000 (0b00000001000x00001x00xx0000000000)

Bit	Reset	Description
31	DISABLE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	NORMAL	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	NORMAL	ASSERT_PLL_RESET: 0 = NORMAL 1 = RST
28	NORMAL	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	NORMAL	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	NORMAL	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	NORMAL	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	TRISTATE	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	FALSE	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	LOW	PIN_B: 0 = LOW 1 = HIGH
21	LOW	PIN_A: 0 = LOW 1 = HIGH
19	UP	SEQUENCE: 0 = UP 1 = DOWN
18	STOP	LANE_SEQ: 0 = STOP 1 = RUN
17	NO	PDPORT: 0 = NO 1 = YES
16	NO	PDPLL: 0 = NO 1 = YES
15	TRUE	HALT: 0 = FALSE 1 = TRUE
13:12	US	WAIT_UNITS: 0 = US 1 = MS

Bit	Reset	Description
		2 = VSYNC
9:0	0x0	WAIT_TIME

24.6.41 SOR_NV_PDISP_SOR_SEQ_INSTC_0

Offset: 0x2e | Byte Offset: 0xb8 | Read/Write: R/W | Reset: 0x01008000 (0b00000001000x00001x00xx0000000000)

Bit	Reset	Description
31	DISABLE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	NORMAL	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	NORMAL	ASSERT_PLL_RESET: 0 = NORMAL 1 = RST
28	NORMAL	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	NORMAL	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	NORMAL	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	NORMAL	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	TRISTATE	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	FALSE	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	LOW	PIN_B: 0 = LOW 1 = HIGH
21	LOW	PIN_A: 0 = LOW 1 = HIGH
19	UP	SEQUENCE: 0 = UP 1 = DOWN
18	STOP	LANE_SEQ: 0 = STOP 1 = RUN
17	NO	PDPORT: 0 = NO 1 = YES
16	NO	PDPLL: 0 = NO 1 = YES

Bit	Reset	Description
15	TRUE	HALT: 0 = FALSE 1 = TRUE
13:12	US	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

24.6.42 SOR_NV_PDISP_SOR_SEQ_INSTD_0

Offset: 0x2f | Byte Offset: 0xbc | Read/Write: R/W | Reset: 0x01008000 (0b00000001000x00001x00xx0000000000)

Bit	Reset	Description
31	DISABLE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	NORMAL	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	NORMAL	ASSERT_PLL_RESET: 0 = NORMAL 1 = RST
28	NORMAL	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	NORMAL	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	NORMAL	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	NORMAL	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	TRISTATE	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	FALSE	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	LOW	PIN_B: 0 = LOW 1 = HIGH
21	LOW	PIN_A: 0 = LOW 1 = HIGH
19	UP	SEQUENCE: 0 = UP 1 = DOWN
18	STOP	LANE_SEQ: 0 = STOP 1 = RUN
17	NO	PDPORT: 0 = NO

Bit	Reset	Description
		1 = YES
16	NO	PDPLL: 0 = NO 1 = YES
15	TRUE	HALT: 0 = FALSE 1 = TRUE
13:12	US	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

24.6.43 SOR_NV_PDISP_SOR_SEQ_INSTE_0

Offset: 0x30 | Byte Offset: 0xc0 | Read/Write: R/W | Reset: 0x01008000 (0b00000001000x00001x00xx0000000000)

Bit	Reset	Description
31	DISABLE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	NORMAL	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	NORMAL	ASSERT_PLL_RESET: 0 = NORMAL 1 = RST
28	NORMAL	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	NORMAL	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	NORMAL	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	NORMAL	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	TRISTATE	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	FALSE	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	LOW	PIN_B: 0 = LOW 1 = HIGH
21	LOW	PIN_A: 0 = LOW 1 = HIGH
19	UP	SEQUENCE: 0 = UP 1 = DOWN

Bit	Reset	Description
18	STOP	LANE_SEQ: 0 = STOP 1 = RUN
17	NO	PDPORT: 0 = NO 1 = YES
16	NO	PDPLL: 0 = NO 1 = YES
15	TRUE	HALT: 0 = FALSE 1 = TRUE
13:12	US	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

24.6.44 SOR_NV_PDISP_SOR_SEQ_INSTF_0

Offset: 0x31 | Byte Offset: 0xc4 | Read/Write: R/W | Reset: 0x01008000 (0b00000001000x00001x00xx0000000000)

Bit	Reset	Description
31	DISABLE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	NORMAL	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	NORMAL	ASSERT_PLL_RESET: 0 = NORMAL 1 = RST
28	NORMAL	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	NORMAL	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	NORMAL	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	NORMAL	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	TRISTATE	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	FALSE	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	LOW	PIN_B: 0 = LOW 1 = HIGH
21	LOW	PIN_A: 0 = LOW

Bit	Reset	Description
		1 = HIGH
19	UP	SEQUENCE: 0 = UP 1 = DOWN
18	STOP	LANE_SEQ: 0 = STOP 1 = RUN
17	NO	PDPORT: 0 = NO 1 = YES
16	NO	PDPLL: 0 = NO 1 = YES
15	TRUE	HALT: 0 = FALSE 1 = TRUE
13:12	US	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

24.6.45 SOR_NV_PDISP_SOR_PWM_DIV_0

These two registers control the optional PWM function for backlight intensity control. The PWM circuit will be driven off the buffered crystal clock. The first register (SOR_PWM_DIV) prescales the crystal clock and sets the resolution range.

DIVIDE

This field defines the period of the PWM output. Note that a value of 0 has the special meaning to disable PWM output.

pwm freq = reference clock / DIVIDE

DUTY_CYCLE

This specifies the duty cycle of the PWM output. In other words, the number of cycles during PWM period (DIVIDE), hardware will assert high (Hi_PERIOD).

CLK_SEL

SOR PWM can be run at pixel clock. This field selects if PWM is run at crystal clock or pixel clock.

SETTING_NEW

Provides the mechanism to trigger a new configuration for both registers (SOR_PWM_DIV and SOR_PWM_CTL). When updating SOR_PWM_DIV, software must be careful to not overwrite an old value in duty cycle unless doing so is desired. Once a state change has been requested, it must be permitted to complete before attempting to trigger another one.

Examples of computation:

If Fs is the desired frequency:

- DIV_DIVIDE = (reference freq / Fs); DIVIDE > 0 and integer.
- DUTY_CYCLE Range : 1 to DIV_DIVIDE
- To get minimum intensity : DUTY_CYCLE = 1
- To get maximum intensity : DUTY_CYCLE = DIV_DIVIDE

Example 1:

For reference clock 27MHz (Xtal)

Desired : 210Hz (Fs)

DIV_DIVIDE = (27000000/210) = 128571.43

Round down to the 128571 causes desired Freq Fs = 210.0007Hz

Round up to the 128572 causes desired Freq Fs = 209.999Hz

Example 2:

For Crystal 120MHz (pixel clock)

Desired : 50000Hz (Fs)

DIV_DIVIDE = (120000000/50000) = 2400

Usage: boot / initialization

Offset: 0x32 | Byte Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:0	0x0	DIVIDE

24.6.46 SOR_NV_PDISP_SOR_PWM_CTL_0

Usage: boot / initialization / normal operation

Offset: 0x33 | Byte Offset: 0xcc | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxx00000000000000000000000)

Bit	Reset	Description
31	DONE	SETTING_NEW: 0 = DONE 1 = PENDING 1 = TRIGGER
30	PCLK	CLKSEL: 0 = PCLK 1 = XTAL
23:0	0x0	DUTY_CYCLE

24.6.47 SOR_NV_PDISP_SOR_VCRCA0_0

Test and Debug Registers

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 for 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.

The VCRC registers for sub-link A

Usage: debug

Offset: 0x34 | Byte Offset: 0xd0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	CRCM
15:0	X	CRCL

24.6.48 SOR_NV_PDISP_SOR_VCRCB0_0

The VCRC registers for sub-link B

Usage: debug

Offset: 0x36 | Byte Offset: 0xd8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	CRCM
15:0	X	CRCL

24.6.49 SOR_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: 0x38 | Byte Offset: 0xe0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	CRCM
15:0	X	CRCL

24.6.50 SOR_NV_PDISP_SOR_CCRCB0_0

The CCRC registers for sub-link B

Usage: debug

Offset: 0x3a | Byte Offset: 0xe8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	CRCM
15:0	X	CRCL

24.6.51 SOR_NV_PDISP_SOR_EDATAA0_0

EDATA has the encoded data for the serial channels TXC, TX3, TX2, TX1, and TX0. For each link, there are up to 40 bits of encoded data. The encoded bits making up the data are as follows:

For TMDS, the encoded data is as follows:

- EDATA[39:0] =
  - {TMDS_CLK_ENC[9:0],
  - TMDS_TX2_ENC[9:0],
  - TMDS_TX1_ENC[9:0],
  - TMDS_TX0_ENC[9:0]}

For LVDS, the encoded data is as follows:

- EDATA[39:0] =
  - {5'b00000,
  - LVDS_CLK_ENC[6:0],
  - LVDS_A3_ENC[6:0],
  - LVDS_A2_ENC[6:0],
  - LVDS_A1_ENC[6:0],
  - LVDS_A0_ENC[6:0]}

The EDATA registers can be written to when any of the trigger bits for capture are set high.

The EDATA registers for sub-link A

Usage: debug

Offset: 0x3c | Byte Offset: 0xf0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL

24.6.52 SOR_NV_PDISP_SOR_EDATAB0_0

The EDATA registers for sub-link B

Usage: debug

Offset: 0x3e | Byte Offset: 0xf8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL

24.6.53 SOR_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: 0x40 | Byte Offset: 0x100 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TX1
15:0	X	TX0

24.6.54 SOR_NV_PDISP_SOR_COUNTB0_0

The count registers for sub-link B.

Usage: debug

Offset: 0x42 | Byte Offset: 0x108 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TX1
15:0	X	TX0

24.6.55 SOR_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: 0x44 | Byte Offset: 0x110 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	RST	VAL: 0 = RST

24.6.56 SOR_NV_PDISP_SOR_DEBUGB0_0

The debug registers for sub-link B

Usage: debug

Offset: 0x46 | Byte Offset: 0x118 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	RST	VAL: 0 = RST

24.6.57 SOR_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 VSYNCs, 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: 0x48 | Byte Offset: 0x120 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:0	RST	VAL: 0 = RST

24.6.58 SOR_NV_PDISP_SOR_MSCHECK_0

The mode switch monitor is used for hardware debug only. To use, the test should set the CTL bit to CLEAR and then to RUN. At the end of the test, the register can be read. The various fields indicate the number of times the following conditions occurred:

- CRC enable went from false to true
- CRC enable went from true to false
- data enable went from false to true
- data enable went from true to false

All counts are 4 bits, so the count will clamp at 15.

Usage: debug

Offset: 0x49 | Byte Offset: 0x124 | Read/Write: R/W | Reset: 0x8000XXXX (0b1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	RUN	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

24.6.59 SOR_NV_PDISP_SOR_XBAR_CTRL_0

This register controls the crossbar between the SOR and the TMDS analog macro.

There are 2 links per SOR and 4/5 TMDS channels per link. Within the same link, the crossbar are fully connected (i.e. each output channel can be connected to any channel). The crossbar output can also be driven to zero by setting XSEL_* to

XSEL_*_ZERO. Also, the two links can be swapped by setting LINK_SWAP to one. This crossbar can also be bypassed by setting BYPASS to one.

Usage: boot / initialization

Offset: 0x4a | Byte Offset: 0x128 | Read/Write: R/W | Reset: 0x8d111a23 (0b10001101000100010001101000100011)

Bit	Reset	Description
31:29	0x4	LINK1_XSEL_4: 7 = ZERO
28:26	0x3	LINK1_XSEL_3: 7 = ZERO
25:23	0x2	LINK1_XSEL_2: 7 = ZERO
22:20	0x1	LINK1_XSEL_1: 7 = ZERO
19:17	0x0	LINK1_XSEL_0: 7 = ZERO
16:14	0x4	LINK0_XSEL_4: 7 = ZERO
13:11	0x3	LINK0_XSEL_3: 7 = ZERO
10:8	0x2	LINK0_XSEL_2: 7 = ZERO
7:5	0x1	LINK0_XSEL_1: 7 = ZERO
4:2	0x0	LINK0_XSEL_0: 7 = ZERO
1	0x1	LINK_SWAP
0	0x1	BYPASS

24.6.60 SOR_NV_PDISP_SOR_XBAR_POL_0

This register controls the polarity of the channels going into the crossbar. Each input channel can be inverted individually by setting the corresponding bit to one.

Usage: boot / initialization

Offset: 0x4b | Byte Offset: 0x12c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	NORMAL	POL_LINK1_4: 0 = NORMAL 1 = INVERT
8	NORMAL	POL_LINK1_3: 0 = NORMAL 1 = INVERT
7	NORMAL	POL_LINK1_2: 0 = NORMAL 1 = INVERT
6	NORMAL	POL_LINK1_1: 0 = NORMAL 1 = INVERT
5	NORMAL	POL_LINK1_0:

Bit	Reset	Description
		0 = NORMAL 1 = INVERT
4	NORMAL	POL_LINK0_4: 0 = NORMAL 1 = INVERT
3	NORMAL	POL_LINK0_3: 0 = NORMAL 1 = INVERT
2	NORMAL	POL_LINK0_2: 0 = NORMAL 1 = INVERT
1	NORMAL	POL_LINK0_1: 0 = NORMAL 1 = INVERT
0	NORMAL	POL_LINK0_0: 0 = NORMAL 1 = INVERT

24.6.61 SOR_NV_PDISP_SOR_DP_LINKCTL0_0

PROGRAMMING NOTE: The SOR_DP registers are doubly indexed. The first index selects the SOR, the second index selects port within the SOR. Some DP SORs will support DPA port and DPB port. The capability bits for each SOR will determine what ports exist. The field NV_PDISP_SOR_DP_LINKCTL_ENABLE is used to enable a specific port on a given OR.

SOR_DP_LINKCTL

ENABLE

- _YES The current DP port is enabled and data will be driven to the associated pins
- _NO The current DP port is not enabled and the port will be powered down.

Each DP SOR has two DP ports, but only one can be active at any given time. Only one of the SOR_DP_LINKCTL_ENABLE fields should be set to _YES. Whichever set of SOR_DP registers has ENABLE set to YES will be the set that controls the DP hardware.

This field must be set to _YES before the first test pattern is sent out for link training. If this field is ever set to _NO, the link must be re-trained before it can be used again.

A certain number of active symbols are metered out every transfer unit to maintain a constant data rate. Since hardware has limited precision to represent the number to active symbols, an error will add up over the line width. TUSIZE is programmed to minimize the error to a value that is close to zero or can be handled by the buffer available in the hardware to sustain the slightly higher or lower data rate than the calculated value.

TUSIZE

Transfer unit size - Valid range 64 to 32

A certain number of active symbols are metered out every transfer unit to maintain a constant data rate. Since hardware has limited precision to represent the number to active symbols, an error will add up over the line width. TUSIZE is programmed to minimize the error to a value that is close to zero or can be handled by the buffer available in the hardware to sustain the slightly higher or lower data rate than the calculated value.

SYNCMODE

- _DISABLE = Link clock and stream clock asynchronous

- _ENABLE = Link clock and stream clock synchronous

This bit is used to specify whether the link clock and pixel clock are synchronous or not. This bit is included in the Main Stream Attribute data. The DP hardware treats all pixel clocks as asynchronous, so this should be left as _DISABLE.

ENHANCED_FRAME

- _DISABLE = Enhanced Framing symbol sequence for BS, SR, CPBS, and CPSR not supported.
- _ENABLE = Enhanced Framing symbol sequence for BS, SR, CPBS, and CPSR supported.

Enhanced Framing is required for HDCP over DP. This should be set to _ENABLE when the SOR is being used for DP and HDCP is enabled.

LANE_COUNT

- _ZERO = no lanes, DP disabled
- _ONE = One lane
- _TWO = Two lanes
- _FOUR = Four lanes

This field controls the lane configuration of the DP datapath. Power controls for individual lanes are in NV_PDISP_SOR_DP_PADCTL. For one-lane configuration, Lane0 is used. For 2-lane configuration, Lane0 and Lane1 are used. Source may choose any lane count as long as it does not exceed the capability of DisplayPort receiver as indicated in the receiver capability field. This field should only be changed when the SOR is asleep.

COMPLIANCE TEST PATTERN

This field is not implemented in Hardware

FORCE IDLE PATTERN

This is a diagnostics bit which when set will force the hardware to send idle pattern on all lanes enabled. By default, the DP hardware will automatically send out the idle pattern if the OR goes to sleep while the link is still trained. This field provides an additional override if needed.

- _NO - don't force idle pattern
- _YES - force idle pattern

Offset: 0x4c | Byte Offset: 0x130 | Read/Write: R/W | Reset: 0x00000100 (0b0xx0xxxxxxx00000x0xxx0x1000000x0)

Bit	Reset	Description
31	NO	FORCE_IDLEPTTRN: 0 = NO 1 = YES
28	NOPATTERN	COMPLIANCEPTTRN: 0 = NOPATTERN 1 = COLORSQARE
20:16	ZERO	LANECOUNT: 0 = ZERO 1 = ONE 3 = TWO 15 = FOUR
14	DISABLE	ENHANCEDFRAME: 0 = DISABLE 1 = ENABLE
10	DISABLE	SYNCMODE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8:2	0x40	TUSIZE
0	NO	ENABLE: 0 = NO 1 = YES

24.6.62 SOR_NV_PDISP_SOR_LANE_DRIVE_CURRENT0_0

SOR_LANE_DRIVE_CURRENT

Transmitter main output drive level. Used to set transmitter main output drive level for each of the four lanes.

bit<7> is dummy

bit<6:3> is decoded into each 3.2mA unit

bit<2:0> are mapped to binary weighted units of 1.6mA, 0.8mA, 0.4mA. Each step is 400uA; the max setting=x100 0111=28mA

- x000 0000 -> 0mA
- ...
- x100 0111 -> 28mA max
- ...
- x111 1111 -> maxed out

Voltage is for single-ended output amplitude with double 50 ohm termination. Voltage is independent of termination control.

This register will be used during link training to set the voltage swing as requested by the sink device. The value to set for the drive current is dependent on the current value of the pre-emphasis in SOR_LANE_PREEMPHASIS. There is a separate LEVEL define for each valid pre-emphasis value..

Note: When DP is active, the LANE0_DP_LANE2 field should be used for lane2 and LANE2_DP_LANE0 should be used for lane0.

Offset: 0x4e | Byte Offset: 0x138 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	LANE3_DP_LANE3: 17 = P0_LEVEL0 21 = P1_LEVEL0 26 = P2_LEVEL0 34 = P3_LEVEL0 26 = P0_LEVEL1 32 = P1_LEVEL1 39 = P2_LEVEL1 34 = P0_LEVEL2 43 = P1_LEVEL2 51 = P0_LEVEL3
23:16	0x0	LANE2_DP_LANE0: 17 = P0_LEVEL0 21 = P1_LEVEL0 26 = P2_LEVEL0 34 = P3_LEVEL0 26 = P0_LEVEL1 32 = P1_LEVEL1 39 = P2_LEVEL1 34 = P0_LEVEL2 43 = P1_LEVEL2 51 = P0_LEVEL3

Bit	Reset	Description
15:8	0x0	LANE1_DP_LANE1: 17 = P0_LEVEL0 21 = P1_LEVEL0 26 = P2_LEVEL0 34 = P3_LEVEL0 26 = P0_LEVEL1 32 = P1_LEVEL1 39 = P2_LEVEL1 34 = P0_LEVEL2 43 = P1_LEVEL2 51 = P0_LEVEL3
7:0	0x0	LANE0_DP_LANE2: 17 = P0_LEVEL0 21 = P1_LEVEL0 26 = P2_LEVEL0 34 = P3_LEVEL0 26 = P0_LEVEL1 32 = P1_LEVEL1 39 = P2_LEVEL1 34 = P0_LEVEL2 43 = P1_LEVEL2 51 = P0_LEVEL3

24.6.63 SOR_NV_PDISP_SOR_LANE4_DRIVE_CURRENT0_0

SOR_LANE4_DRIVE_CURRENT

Non-DP SORs contain five total lanes; four data lanes and one clock. There was not enough room in one register to fit all five lanes. LANE4 controls the fifth lane.

Offset: 0x50 | Byte Offset: 0x140 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	LANE4

24.6.64 SOR_NV_PDISP_SOR_LANE_PREEMPHASIS0_0

SOR_LANE_PREEMPHASIS

Transmitter main output pre-emphasis for each of the four lanes.

bit<7> is dummy

bit<6:3> is decoded into each 1.6mA unit

bit<2:0> is mapped to binary weighted units of 0.8mA, 0.4mA, 0.2mA. Each step is 200 µA; the maximum setting=x010 1111 = 9.4mA, if the main driver is not borrowing; the main driver has higher priority to borrow.

- x000 0000 -> 0mA
- ...
- x010 1111 -> 9.4mA max
- ...
- x111 1111 -> maxed out

This register will be used during link training to set the pre-emphasis level of each lane as requested by the sink device. The sink device may request this value to be changed during Training Pattern 1 or Training Pattern 2 during link training.

The pre-emphasis value for a given level is dependent on the drive current setting.

Note: when DP is active, the LANE0_DP_LANE2 field should be used for lane2 and LANE2_DP_LANE0 should be used for lane0.

Offset: 0x52 | Byte Offset: 0x148 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	LANE3_DP_LANE3: 0 = D0_LEVEL0 0 = D1_LEVEL0 0 = D2_LEVEL0 0 = D3_LEVEL0 4 = D0_LEVEL1 6 = D1_LEVEL1 17 = D2_LEVEL1 8 = D0_LEVEL2 13 = D1_LEVEL2 17 = D0_LEVEL3
23:16	0x0	LANE2_DP_LANE0: 0 = D0_LEVEL0 0 = D1_LEVEL0 0 = D2_LEVEL0 0 = D3_LEVEL0 4 = D0_LEVEL1 6 = D1_LEVEL1 17 = D2_LEVEL1 8 = D0_LEVEL2 13 = D1_LEVEL2 17 = D0_LEVEL3
15:8	0x0	LANE1_DP_LANE1: 0 = D0_LEVEL0 0 = D1_LEVEL0 0 = D2_LEVEL0 0 = D3_LEVEL0 4 = D0_LEVEL1 6 = D1_LEVEL1 17 = D2_LEVEL1 8 = D0_LEVEL2 13 = D1_LEVEL2 17 = D0_LEVEL3
7:0	0x0	LANE0_DP_LANE2: 0 = D0_LEVEL0 0 = D1_LEVEL0 0 = D2_LEVEL0 0 = D3_LEVEL0 4 = D0_LEVEL1 6 = D1_LEVEL1 17 = D2_LEVEL1 8 = D0_LEVEL2 13 = D1_LEVEL2 17 = D0_LEVEL3

24.6.65 SOR_NV_PDISP_SOR_LANE4_PREEMPHASIS0_0

SOR_LANE4_PREEMPHASIS

Transmitter main output pre-emphasis for lane4 of each link.

Offset: 0x54 | Byte Offset: 0x150 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	LANE4

24.6.66 SOR_NV_PDISP_SOR_POSTCURSOR0_0

SOR_LANE_POSTCURSOR

Controls Post-cursor2 amplitude for each lane. This is required during link training at HBR2 speeds.

This should be set to 0 for RBR or HBR training.

bit<7:5> is dummy

bit<4:3> is decoded into each 1.6mA unit

bit<2:0> is mapped to binary weighted units of 0.8mA, 0.4mA, 0.2mA. Each step is 200 µA; the maximum setting=xxx1 1111 = 6.2mA, if the main driver is not borrowing; the main driver has higher priority to borrow.

- xxx0 0000 -> 0mA
- ...
- xxx1 1111 -> 6.2mA max

Offset: 0x56 | Byte Offset: 0x158 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	LANE3_DP_LANE3
23:16	0x0	LANE2_DP_LANE0
15:8	0x0	LANE1_DP_LANE1
7:0	0x0	LANE0_DP_LANE2

24.6.67 SOR_NV_PDISP_SOR_DP_CONFIG0_0

SOR_DP_CONFIG

This register contains various other controls which affect the behavior of the DP logic.

WATERMARK

This field defines the number of symbols that the DP logic will wait for at the beginning of a line before ending blanking. This number will be based on the pixel rate, the link speed, bits per pixel, tusize and line active width.

IDLE_BEFORE_ATTACH

When a head is attached to the SOR while in DisplayPort mode, the OR will send out five idle patterns before acknowledging the attach command. The DP spec requires that at least five idle patterns are sent before changing any timing parameters. This give the hardware time to calculate a new M value if the pclk frequency changed.

- _ENABLE: wait 5 idle patterns after attach before acknowledging it
- _DISABLE: acknowledge attach right away.

This register is for debug purposes only; it should be set to _ENABLE all the time for production.

RD_RESET_VAL

When training pattern 2 begins, the running disparity of the 8B10B encoding is reset. The DP spec defines that the disparity of the first symbol sent out should be negative. By resetting the 8B10B running disparity to positive, the first symbol sent out will have a negative disparity.

- _POSITIVE: The internal running disparity is reset to positive, so the first 10b symbol sent out will have negative disparity.

- _NEGATIVE: The internal running disparity is reset to negative, so the first 10b symbol send out will have positive disparity.

This value is not expected to change unless we encounter a panel that expects a different disparity.

ACTIVESYM_COUNT, ACTIVESYM_FRAC and ACTIVESYM_POLARITY:

These three fields control the number of active symbols sent out every transfer unit. Hardware maintains a TU count which is used for metering. ACTIVESYM_COUNT is the number of symbols sent out every transfer irrespective of the tu count. Based on the ACTIVESYM_POLARITY which can be 0 or 1, we can send extra symbol for a particular TU based on the ACTIVESYM_FRAC (active fraction). If active polarity is 0 and ACTIVESYM_FRAC is say 5, hardware send an extra symbols every 5th TU. If active polarity is 1 and ACTIVESYM_FRAC is 5, hardware will send an extra symbols every TU except for the 5th TU.

For instance if the number of active symbols per TU is calculated to be 17.2, ACTIVESYM_COUNT is set to 17. Fractional value is 0.2 = 1/5 which translates to 1 symbol every 5 TUs. This translates to ACTIVESYM_FRAC of 5 and ACTIVESYM_POLARITY of 0.

For instance if the number of active symbols per TU is calculated to be 17.8, ACTIVESYM_COUNT is set to 17. Fractional value is 0.8 = 4/5 which translates to 1 symbol every TU except for the 5th TU. This translates to ACTIVESYM_FRAC of 5 and ACTIVESYM_POLARITY of 1.

ACTIVESYM_COUNT, ACTIVESYM_FRAC and ACTIVESYM_POLARITY:

These three fields control the number of active symbols sent out every transfer unit. Hardware maintains a TU count which is used for metering. ACTIVESYM_COUNT is the number of symbols sent out every transfer irrespective of the tu count. Based on the ACTIVESYM_POLARITY which can be 0 or 1, we can send extra symbol for a particular TU based on the ACTIVESYM_FRAC (active fraction). If active polarity is 0 and ACTIVESYM_FRAC is say 5, hardware send an extra symbols every 5th TU. If active polarity is 1 and ACTIVESYM_FRAC is 5, hardware will send an extra symbols every TU except for the 5th TU.

For instance if the number of active symbols per TU is calculated to be 17.2, ACTIVESYM_COUNT is set to 17. Fractional value is 0.2 = 1/5 which translates to 1 symbol every 5 TU's. This translates to ACTIVESYM_FRAC of 5 and ACTIVESYM_POLARITY of 0.

For instance if the number of active symbols per TU is calculated to be 17.8, ACTIVESYM_COUNT is set to 17. Fractional value is 0.8 = 4/5 which translates to 1 symbol every TU except for the 5th TU. This translates to ACTIVESYM_FRAC of 5 and ACTIVESYM_POLARITY of 1.

Offset: 0x58 | Byte Offset: 0x160 | Read/Write: R/W | Reset: 0x94000000 (0b1xx1x1x0xxxx0000x0000000xx000000)

Bit	Reset	Description
31	NEGATIVE	RD_RESET_VAL: 0 = POSITIVE 1 = NEGATIVE
28	ENABLE	IDLE_BEFORE_ATTACH: 0 = DISABLE 1 = ENABLE
26	ENABLE	ACTIVESYM_CNTL: 0 = DISABLE 1 = ENABLE
24	NEGATIVE	ACTIVESYM_POLARITY: 0 = NEGATIVE 1 = POSITIVE
19:16	0x0	ACTIVESYM_FRAC
14:8	0x0	ACTIVESYM_COUNT

Bit	Reset	Description
5:0	0x0	WATERMARK

24.6.68 SOR_NV_PDISP_SOR_DP_MN0_0

SOR_DP_MN

This register contains controls to tweak the values used for M and N in the Main Stream Attribute data. These controls are for debugging purposes and possible workarounds. See Section 2.3.3 of the DisplayPort specification for more detailed descriptions of M and N.

N_VAL

This is the value that will be used for calculating M. This value is not used in the Main Stream Attributes. Main Stream attributes always send 2^15 for Nvid. This should not need to be programmed beyond its default value. If this value needs to be changed, it should be done while the OR is detached. Otherwise, the calculated M value will be incorrect for some time, or until the next mode switch.

M_DELTA

This defines an offset that will be used to modify the calculated M value. This should not need to be programmed beyond its default value.

M_MOD

Describes how the M_DELTA field should be applied to the M value.

- _NONE: The M_DELTA is ignored and not used.
- _INC: The M_DELTA value is added to M.
- _DEC: The M_DELTA value is subtracted from M.

This should not need to be programmed beyond its default value.

Offset: 0x5a | Byte Offset: 0x168 | Read/Write: R/W | Reset: 0x00008000 (0b00xx00000000000010000000000000000)

Bit	Reset	Description
31:30	NONE	M_MOD: 0 = NONE 1 = INC 2 = DEC
27:24	0x0	M_DELTA
23:0	0x8000	N_VAL

24.6.69 SOR_NV_PDISP_SOR_DP_PADCTL0_0

SOR_DP_PADCTL

This register directly controls the DP pads. Some values in this register are only valid in DP mode when NV_PDISP_SOR_DP_LINKCTL_ENABLE == _YES.

PD_TXD_

Individual controls to power down the lanes. NV_PDISP_SOR_DP_LINKCTL_LANE_COUNT sets the number of lanes that are active in the DP datapath. These fields control power to the macro pins. When reducing the number of active lanes, the inactive lanes should be powered down before the lane configuration is changed. See section 5.1.2 of the DP Specification.

If a lane is powered down, link training will need to be done before that lane can be used again.

- _YES: Force the lane to power down
- _NO: Enable power to the lane. Do not power down.

Only valid in DP mode when NV_PDISP_SOR_DP_LINKCTL_ENABLE == _YES.

COMMONMODE_TXD_

This is used to force the lanes to output the common mode voltage. This is required before link training begins. Section 3.5.3.3 of the DP specification dictates that the lanes must be precharged to the common mode voltage for at least 10 microseconds before beginning link training. Please see the DP specification for more information:

Note: when DP is active, the TXD_0_DP_TXD_2 field should be used for lane2 and TXD_2_DP_TXD_0 should be used for lane0.

- _DISABLE: Normal operation
- _ENABLE: Force the lanes to output the common mode voltage

TX_PU_VALUE

TX pull-up current source drive

- x000xxxx -> 0mA
- x001xxxx -> 1mA
- x010xxxx -> 2mA
- x011xxxx -> 3mA
- x100xxxx -> 4mA
- x101xxxx -> 5mA
- x110xxxx -> 6mA

TX_PU

- _ENABLE: Pull-up current sources enabled
- _DISABLE: Pull-up current sources disabled

VCMMODE

VCM pin mode select

- _TRISTATE:tristate (default)
- _TEST_MUX:under test-mux control
- _WEAK_PULLDOWN:0 V weak pull-down
- _STRONG_PULLDOWN: 0V strong pull-down

This is a test mode pin, it does not need to be programmed by software.

REG_CTRL

Internal regulator control. These are spare input bits to the analog macro that have no defined function currently.

PAD_CAL_PD

Powerdown control for the pad calibration logic.

- _POWERDOWN: Power down
- _POWERUP:Normal operation

VCO_2X

This is a VCO startup diagnostics control input to the DP analog macro.

- _ENABLE: forced VCO oscillation mode
- _DISABLE: normal VCO operation

SPARE

Spare PLL control bits.

This field maps to ports SPAREPLL_1 to SPAREPLL_7 of the DP/LVDS macro. Some bits in this field are used only for debug. Some bits are used for VCO Gain Calibration and LOADADJ calibration.

The field names used below are the names of the ports on the macro.

SPARE_1: Maps to "vcolimit_sel":

- when vcolimit_sel=0, the clamp counter counts 128 cycles <-> 474ns @270Mhz;
- when vcolimit_sel=1, the clamp counter counts 32 cycles <-> 118ns @ 270MHz.

SPARE_2: Maps to "loadadj_sync_en":

- when loadadj_sync_en=0,the loadadj-calibration code is reset to 0000;
- when loadadj_sync_en=1,the loadadj-calibration code is swept and the best value is used.
- Used during LOADADJ calibration. Must be set to 0 and then to 1 to trigger the calibration sequence.

SPARE_3: Maps to "loadadj_bypn":

- when loadadj_bypn=0, the loadadj-calibration block is bypassed;
- when loadadj_bypn=1, the loadadj-calibration block is being used.
- Used during LOADADJ calibration to select the auto-cal value instead of the register value.

SPARE_4: Maps to "vcocalib_enb":

- when vcocalib_enb=0,enable vco-calibration, sweep the vcogain<3:0> code and the best value is used.
- when vcocalib_enb=1, reset the vco-calibration block.
- Used during VCO Gain Calibration. Must be set to 1 and then back to 0 to trigger the calibration sequence.

SPARE_5: Maps to "vcocalib_overwrb":

- when vcocalib_overwrb=0, overwrite/disable the vco-calibration block;
- when vcocalib_overwrb=1, enable the vco-calibration block.
- Used during VCO Gain Calibration.

SPARE_6: Maps to "vcocalib_ts0":

- when vcocalib_ts0=0, vco-calibration block counts 96 cycles <-> 355ns @270MHz;
- when vcocalib_ts0=1, vco-calibration block counts 128 cycles <-> 474ns @270Mhz

SPARE_7: Maps to "vcolimit_disable":

- when vcolimit_disable=0, enable the VCO-protection clamp;
- when vcolimit_disable=1, disable the VCO-protection clamp.

Offset: 0x5c | Byte Offset: 0x170 | Read/Write: R/W | Reset: 0x00800000 (0b00000000100000000000000000000000)

Bit	Reset	Description
31:25	0x0	SPARE
24	DISABLE	VCO_2X: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
23	POWERDOWN	PAD_CAL_PD: 0 = POWERUP 1 = POWERDOWN
22	DISABLE	TX_PU: 0 = DISABLE 1 = ENABLE
21:20	0x0	REG_CTRL
19:16	TRISTATE	VCMMODE: 0 = TRISTATE 1 = TEST_MUX 2 = WEAK_PULLDOWN 4 = STRONG_PULLDOWN
15:8	0x0	TX_PU_VALUE
7	DISABLE	COMMONMODE_TXD_3_DP_TXD_3: 0 = DISABLE 1 = ENABLE
6	DISABLE	COMMONMODE_TXD_2_DP_TXD_0: 0 = DISABLE 1 = ENABLE
5	DISABLE	COMMONMODE_TXD_1_DP_TXD_1: 0 = DISABLE 1 = ENABLE
4	DISABLE	COMMONMODE_TXD_0_DP_TXD_2: 0 = DISABLE 1 = ENABLE
3	YES	PD_TXD_3: 0 = YES 1 = NO
2	YES	PD_TXD_0: 0 = YES 1 = NO
1	YES	PD_TXD_1: 0 = YES 1 = NO
0	YES	PD_TXD_2: 0 = YES 1 = NO

24.6.70 SOR_NV_PDISP_SOR_DP_DEBUG0_0

SOR_DP_DEBUG register is meant for debug purposes. It indicates of underflows or overflows at various point in the main link datapath.

- _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.

LANE0/1/2/3_FIFO_UNDERFLOW

When active symbol control is enabled (controlled by NV_PDISP_SOR_DP_CONFIG_ACTIVESYM_CNTL), a programmed value of active symbols are transmitted every transfer unit. This error is set when active symbol control is enabled and lane FIFOs underflow.

LANE0/1/2/3_PIXPACK_OVERFLOW

This error indicates that pixel packing registers have overflowed. This would happen if the incoming data rate (at pixel clock) is much faster than outgoing data rate (at link clock).

LANE0/1/2/3_STEER_ERROR

These bits indicate that an unexpected error has occurred in the steering logic causing imbalance in the number of bits in the lanes.

SPKT_OVERRUN

If the watermark is attained while a secondary packet is still being scanned out, then this bit will be set. All secondary packets should complete before Blank End needs to be set.

LANE0/1/2/3_FIFO_OVERFLOW

This error is set if the lane FIFOs overflow. This error would point to incorrect programming of tusize, active sym parameters. These params control the rate at which the lane FIFOs are read.

If these params are smaller than expected values, symbols will keep accumulating in the lane FIFO during the course of a horizontal line and would eventually cause an overflow.

Offset: 0x5e | Byte Offset: 0x178 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000000000000)

Bit	Reset	Description
16	NO	LANE3_FIFO_OVERFLOW:w1c 0 = NO 1 = YES 1 = RST
15	NO	LANE2_FIFO_OVERFLOW:w1c 0 = NO 1 = YES 1 = RST
14	NO	LANE1_FIFO_OVERFLOW:w1c 0 = NO 1 = YES 1 = RST
13	NO	LANE0_FIFO_OVERFLOW:w1c 0 = NO 1 = YES 1 = RST
12	NO	SPKT_OVERRUN:w1c 0 = NO 1 = YES 1 = RST
11	NO	LANE3_STEER_ERROR:w1c 0 = NO 1 = YES 1 = RST
10	NO	LANE2_STEER_ERROR:w1c 0 = NO 1 = YES 1 = RST
9	NO	LANE1_STEER_ERROR:w1c 0 = NO 1 = YES 1 = RST
8	NO	LANE0_STEER_ERROR:w1c 0 = NO 1 = YES 1 = RST

Bit	Reset	Description
7	NO	LANE3_PIXPACK_OVERFLOW:w1c 0 = NO 1 = YES 1 = RST
6	NO	LANE2_PIXPACK_OVERFLOW:w1c 0 = NO 1 = YES 1 = RST
5	NO	LANE1_PIXPACK_OVERFLOW:w1c 0 = NO 1 = YES 1 = RST
4	NO	LANE0_PIXPACK_OVERFLOW:w1c 0 = NO 1 = YES 1 = RST
3	NO	LANE3_FIFO_UNDERFLOW:w1c 0 = NO 1 = YES 1 = RST
2	NO	LANE2_FIFO_UNDERFLOW:w1c 0 = NO 1 = YES 1 = RST
1	NO	LANE1_FIFO_UNDERFLOW:w1c 0 = NO 1 = YES 1 = RST
0	NO	LANE0_FIFO_UNDERFLOW:w1c 0 = NO 1 = YES 1 = RST

24.6.71 SOR_NV_PDISP_SOR_DP_SPARE0_0

NV_PDISP_SOR_DP_SPARE

This is an extra register in the DP SORs that can be used for future features

SEQ_ENABLE

Used to enable the sequencer in DP mode. This needs to be set for eDP where we need to run sequencer to control backlight.

PANEL

Specify whether the DP panel is external or internal (notebook panel)

- EXTERNAL: An external DP panel is connected, normal operation status will report an internal link, and the scrambler will reset to 0xFFFE instead of 0xFFFF

SOR_CLK_SEL

This is used for SOR testbench. Specify whether to use a safe clock or the macro clock as the SOR clock.

SAFE_SORCLK: Use the safe clock as the SOR clock

MACRO_SORCLK: Use the macro clock as the SOR clock

Offset: 0x60 | Byte Offset: 0x180 | Read/Write: R/W | Reset: 0x00000002 (0b00000000000000000000000000000010)

Bit	Reset	Description
31:3	0x0	REG
2	SAFE_SORCLK	SOR_CLK_SEL: 0 = SAFE_SORCLK 1 = MACRO_SORCLK
1	INTERNAL	PANEL: 0 = EXTERNAL 1 = INTERNAL
0	NO	SEQ_ENABLE: 0 = NO 1 = YES

24.6.72 SOR_NV_PDISP_SOR_DP_AUDIO_CTRL_0

SOR_DP_AUDIO_CTRL

ENABLE

This is the global enable/disable field for Audio over DisplayPort. When set to _NO, no audio stream packets, audio timestamp packets, or audio infoframes will be sent. When set to _YES, audio stream packets will be sent whenever audio arrives, and timestamp and infoframe packets will be sent once per frame during hblank.

MUTE

This field controls the AudioMute_Flag in the VB-ID. When muted, audio samples sent to the sink device are not played. This is an override to the audio mute signal coming from the Azalia Codec.

- DISABLE: The mute is not asserted, audio will be audible
- ENABLE: The mute is asserted. The sink device should not play any audio samples that are sent. If SW changes MUTE from ENABLE to AUTO, hardware will take control of unmuting after the new audio infoframe and audio timestamp have been sent.
- AUTO: The mute signal is controlled by hardware. Audio is muted during audio format or audio timing change and unmuted after the new audio infoframe and audio timestamp have been sent.

AUDIO INFOFRAME HEADER_OVERRIDE

HB1-HB3 are fixed values as defined by the DisplayPort specification. This field will let the values in NV_PDISP_HDMI_AUDIO_INFOFRAME_HEADER override the default values. This can be used for Debugging purposes and can be left at DISABLE for production.

- DISABLE: Use the default values of HB0=0x00, HB1=0x84, HB2=0x1B, HB3=0x44
- ENABLE: Replace the contents of the Audio Infoframe Header with the values in NV_PDISP_HDMI_AUDIO_INFOFRAME_HEADER

GENERIC_INFOFRAME_ENABLE

This field allows software to send infoframes (AVI, etc.) other than audio infoframe. When enabled NV_PDISP_SOR_DP_GENERIC_INFOFRAME_HEADER, NV_PDISP_SOR_DP_GENERIC_INFOFRAME_SUBPACK* priv regs is used as infoframe header and packet data. The infoframe is sent once per vertical blanking period.

PACKETID

All secondary data packets have a packet ID in the header. All packets that are associated with the same audio stream need to have the same packet ID. This field is used to set that ID. This field can be used for debugging purposes and can be left at 0 for production.

CT_SELECT (Coding Type), CC_SELECT (Channel Count), SF_SELECT (Sampling Frequency), SS_SELECT (Sample Size), CA_SELECT (Channel/Speaker Allocation)

These fields control the values of several fields in the Audio Infoframe. The Azalia codec will detect these values, and can automatically place the correct values in the Infoframe.

- SW: The Audio Infoframe will contain the value from NV_PDISP_HDMI_AUDIO_INFOFRAME_SUBPACK0*
- HW: The Audio Infoframe will contain the value set by the Azalia codec.

NEW_SETTINGS

All the fields/settings in this register will not take effect until NEW_SETTINGS is set to _TRIGGER. When the new settings are active, this field will automatically be set to _DONE. If hardware is sending an audio packet when this is set to trigger, then it will wait for the completion of the packet before the new settings are activated. Hence the max delay from TRIGGER to DONE will be time to complete an audio packet (56* (1/162 * 10^6) = 0.35 µs).

Offset: 0x62 | Byte Offset: 0x188 | Read/Write: R/W | Reset: 0x00Xf0001 (0b0xxxxxxxxxx111110000000000xx00x1)

Bit	R/W	Reset	Description
31	RW	DONE	NEW_SETTINGS:w1c 0 = DONE 1 = PENDING 1 = TRIGGER
21	RO	X	MUTE_STATUS: 0 = DISABLE 1 = ENABLE
20	RW	HW	CA_SELECT: 0 = SW 1 = HW
19	RW	HW	SS_SELECT: 0 = SW 1 = HW
18	RW	HW	SF_SELECT: 0 = SW 1 = HW
17	RW	HW	CC_SELECT: 0 = SW 1 = HW
16	RW	HW	CT_SELECT: 0 = SW 1 = HW
15:8	RW	0x0	PACKET_ID
7	RW	NO	GENERIC_INFOFRAME_ENABLE: 0 = NO 1 = YES
6	RW	DISABLE	INFOFRAME_HEADER_OVERRIDE: 0 = DISABLE 1 = ENABLE
3:2	RW	AUTO	MUTE: 0 = AUTO 1 = DISABLE 2 = ENABLE
0	RW	YES	ENABLE: 0 = NO 1 = YES

24.6.73 SOR_NV_PDISP_SOR_DP_AUDIO_HBLANK_SYMBOLS_0

SOR_DP_AUDIO_HBLANK_SYMBOLS

The packet generation logic needs to know the length of the hblank period. If there is no room in the current hblank for a new packet, it will be delayed until the next blanking period. This field should be programmed during the second Supervisor interrupt based on the new raster dimensions.

Y = 12/number of lanes

number of symbols/hblank = ((SetRasterBlankEnd.X + SetRasterSize.Width - SetRasterBlankStart.X - 7) * link_clk / pclk) - 3*enhanced_framing - Y

Y accounts for the time required to send BS, VBID, Mvid, Maud.

The value 7 is subtracted to account for reduction in hblank interval in the link clock domain due to uncertainties arising due to pixels crossing async boundary from pixel clock (rg_clk) to link clock (or_clk). Since we use two async FIFOs to transfer pixels from rg_clk to orclk, the uncertainties are effectively doubled. The value has been experimentally determined.

The following formulas can be used to calculate the maximum audio sampling rate that can be supported by DisplayPort given the current raster dimensions. DisplayPort has much more bandwidth during blanking periods than HDMI has, so hblank size is less of an issue.

- number of free symbols/hblank = ((SetRasterBlankEnd.X + SetRasterSize.Width - SetRasterBlankStart.X - 7) * link_clk / pclk - 2 - (3*enhanced_framing)) * #lanes - 12
- Size of a packet for 2ch audio = 20 symbols (up to 2 samples)
- Size of a packet for 8ch audio = 40 symbols
- Size of an audio packet header plus control symbols = 2*#lanes + 8 symbols (assuming < 32 samples per line)
- number of packets/hblank for 2ch audio = Floor ((number of free symbols/hblank - (2*#lanes + 8) / 20)
- number of packets/hblank for 8ch audio = Floor ((number of free symbols/hblank - (2*#lanes + 8) / 40)

Maximum audio sample rate possible:

- number of audio samples/line = SetRasterSize.Width * audio_fs / pclk
- number of audio packets needed for 2ch audio = Ceiling(SetRasterSize.Width * audio_fs / (pclk*2))
- number of audio packets needed for 3-8ch audio = SetRasterSize.Width * audio_fs / pclk

If the number of audio packets needed > number of packets/hblank, then that audio frequency is not supported.

Offset: 0x63 | Byte Offset: 0x18c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000000000000)

Bit	Reset	Description
16:0	0x0	VALUE

24.6.74 SOR_NV_PDISP_SOR_DP_AUDIO_VBLANK_SYMBOLS_0

SOR_DP_AUDIO_VBLANK_SYMBOLS

The packet generation logic needs to know the length of a line during VBLANK. If there is no room in the current line for a new packet, it will be delayed until the next line. This field should be programmed during the second Supervisor interrupt based on the new raster dimensions.

Y accounts for the time required to send main stream attributes.

25 accounts for reduction of the vblank interval in the link clock domain due to active region transmission spilling on to the vblank region. Note this happen only on the first vblank line while transitioning from active to blanking. This value is

experimentally determined. This might need some adjustment if NV_PDISP_SOR_AUDIO_DEBUG_BLANK_COUNT_ERROR is triggered. Note that this value will not exceed TUSIZE.

Y = (36/number of lanes) + 3;

number of symbols/line in vblank = ((SetRasterBlankStart.X - SetRasterBlankEnd.X - 25) * link_clk / pclk) - Y - 1

Offset: 0x64 | Byte Offset: 0x190 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
20:0	0x0	VALUE

24.6.75 SOR_NV_PDISP_SOR_DP_GENERIC_INFOFRAME_HEADER_0

SOR_DP_GENERIC_INFOFRAME_HEADER

This register should be written with the value of the DP InfoFrame header (infoframe other than audio infoframe).

Offset: 0x65 | Byte Offset: 0x194 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	HB3
23:16	0x0	HB2
15:8	0x0	HB1
7:0	0x0	HB0

24.6.76 SOR_NV_PDISP_SOR_DP_GENERIC_INFOFRAME_SUBPACK0_0

SOR_DP_GENERIC_INFOFRAME_SUBPACK0

This register should be written with the bytes of the DP Infoframe packet data ie DB0- DB27 in figure 2-21 of the DP specification, v1.1a.

Offset: 0x66 | Byte Offset: 0x198 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DB3
23:16	0x0	DB2
15:8	0x0	DB1
7:0	0x0	DB0

24.6.77 SOR_NV_PDISP_SOR_DP_GENERIC_INFOFRAME_SUBPACK1_0

Offset: 0x67 | Byte Offset: 0x19c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DB7
23:16	0x0	DB6
15:8	0x0	DB5
7:0	0x0	DB4

24.6.78 SOR_NV_PDISP_SOR_DP_GENERIC_INFOFRAME_SUBPACK2_0

Offset: 0x68 | Byte Offset: 0x1a0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DB11
23:16	0x0	DB10
15:8	0x0	DB9
7:0	0x0	DB8

24.6.79 SOR_NV_PDISP_SOR_DP_GENERIC_INFOFRAME_SUBPACK3_0

Offset: 0x69 | Byte Offset: 0x1a4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DB15
23:16	0x0	DB14
15:8	0x0	DB13
7:0	0x0	DB12

24.6.80 SOR_NV_PDISP_SOR_DP_GENERIC_INFOFRAME_SUBPACK4_0

Offset: 0x6a | Byte Offset: 0x1a8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DB19
23:16	0x0	DB18
15:8	0x0	DB17
7:0	0x0	DB16

24.6.81 SOR_NV_PDISP_SOR_DP_GENERIC_INFOFRAME_SUBPACK5_0

Offset: 0x6b | Byte Offset: 0x1ac | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DB23
23:16	0x0	DB22
15:8	0x0	DB21
7:0	0x0	DB20

24.6.82 SOR_NV_PDISP_SOR_DP_GENERIC_INFOFRAME_SUBPACK6_0

Offset: 0x6c | Byte Offset: 0x1b0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DB27
23:16	0x0	DB26
15:8	0x0	DB25
7:0	0x0	DB24

24.6.83 SOR_NV_PDISP_SOR_DP_TPG_0

NV_PDISP_SOR_DP_TPG

This register is used to control the training patterns needed during link training. This should be set after the SOR clock has been set up and the lanes have been precharged.

There is one field per lane. If the same test pattern is to be sent on all lanes, they must all be switched to that pattern at the same time in order to stay in sync

Link Training Pattern Setting

- _NOPATTERN Training not in progress (or disabled)
- _TRAINING1 Training Pattern 1
- _TRAINING2 Training Pattern 2
- _TRAINING3 Training Pattern 3
- _D102 D10.2 test pattern (unscrambled) transmitted (same as Training Pattern 1)
- _SBLEERRATE Symbol Error Rate measurement pattern transmitted
- _PRBS7 PRBS7 transmitted
- _CSTM Custom 80 bit test pattern defined in NV_PDISP_SOR_DP_LQ_CSTM
- _HBR2_COMPLIANCE HBR2 Compliance Eye pattern. Repetition of scrambled 0s before 8b10b encoding. Sends SR periodically as defined in NV_PDISP_SOR_DP_TPG_CONFIG_HBR2_COMPLIANCE_PERIOD

This is used for HBR2 electrical compliance.

SCRAMBLEREN

- _DISABLE- DisplayPort transmitter disables scrambler
- _ENABLE_GALIOS - DisplayPort transmitter scrambles data symbols before transmission
- _ENABLE_FIBONACCI - This enables a different type of scrambler. This will probably not be used.

CHANNEL_CODING

This bit set to _ENABLE when DisplayPort receiver supports the Main Link channel coding specification as specified in ANSI X3.230-1994, clause 11. This is reserved for future expansion if a different coding scheme is used later.

See section 2.9.3.6 for more information about the different test patterns. For convenience, table 2-76 is provided here to specify the required SCRAMBLEREN and CHANNELCODING values for each pattern.

```
                | Requires        | Requres
Pattern         | Channel Coding? | Scrambling?
----------------+-----------------+------------
TRAINING1       | Yes             | No
TRAINING2       | Yes             | No
TRAINING3       | Yes             | No
D102            | Yes             | No
SBLERRATE       | Yes             | Yes
PRBS7           | No              | No
CSTM            | No              | No
HBR2_COMPLIANCE | Yes             | Yes
```

Offset: 0x6d | Byte Offset: 0x1b4 | Read/Write: R/W | Reset: 0x50505050 (0bx1010000x1010000x1010000x1010000)

Bit	Reset	Description
30	ENABLE	LANE3_CHANNELCODING: 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
29:28	ENABLE_GALIOS	LANE3_SCRAMBLEREN: 0 = DISABLE 1 = ENABLE_GALIOS 2 = ENABLE_FIBONACCI
27:24	NOPATTERN	LANE3_PATTERN: 0 = NOPATTERN 1 = TRAINING1 2 = TRAINING2 3 = TRAINING3 4 = D102 5 = SBLERRRATE 6 = PRBS7 7 = CSTM 8 = HBR2_COMPLIANCE
22	ENABLE	LANE2_CHANNELCODING: 0 = DISABLE 1 = ENABLE
21:20	ENABLE_GALIOS	LANE2_SCRAMBLEREN: 0 = DISABLE 1 = ENABLE_GALIOS 2 = ENABLE_FIBONACCI
19:16	NOPATTERN	LANE2_PATTERN: 0 = NOPATTERN 1 = TRAINING1 2 = TRAINING2 3 = TRAINING3 4 = D102 5 = SBLERRRATE 6 = PRBS7 7 = CSTM 8 = HBR2_COMPLIANCE
14	ENABLE	LANE1_CHANNELCODING: 0 = DISABLE 1 = ENABLE
13:12	ENABLE_GALIOS	LANE1_SCRAMBLEREN: 0 = DISABLE 1 = ENABLE_GALIOS 2 = ENABLE_FIBONACCI
11:8	NOPATTERN	LANE1_PATTERN: 0 = NOPATTERN 1 = TRAINING1 2 = TRAINING2 3 = TRAINING3 4 = D102 5 = SBLERRRATE 6 = PRBS7 7 = CSTM 8 = HBR2_COMPLIANCE
6	ENABLE	LANE0_CHANNELCODING: 0 = DISABLE 1 = ENABLE
5:4	ENABLE_GALIOS	LANE0_SCRAMBLEREN: 0 = DISABLE 1 = ENABLE_GALIOS 2 = ENABLE_FIBONACCI
3:0	NOPATTERN	LANE0_PATTERN: 0 = NOPATTERN 1 = TRAINING1 2 = TRAINING2

Bit	Reset	Description
		3 = TRAINING3 4 = D102 5 = SBLERRRATE 6 = PRBS7 7 = CSTM 8 = HBR2_COMPLIANCE

24.6.84 SOR_NV_PDISP_SOR_DP_TPG_CONFIG_0

NV_PDISP_SOR_DP_TPG_CONFIG

Additional controls for the DP Test Pattern Generator.

HBR2_COMPLIANCE_PERIOD

The HBR2 compliance pattern is a series of 0s that are scrambled and 8b10b encoded. Periodically, the Scrambler Reset Sequence is sent to keep the source and sink in sync. This field controls how many total symbols there are between the start of a new Scrambler Reset Sequence. This will be defined by a DPCD register on the sink test equipment.

Offset: 0x6e | Byte Offset: 0x1b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000000000000)

Bit	Reset	Description
16:0	0x0	HBR2_COMPLIANCE_PERIOD

24.6.85 SOR_NV_PDISP_SOR_DP_LQ_CSTM0_0

NV_PDISP_SOR_DP_LQ_CSTM

These registers are used to program a custom 80-bit test pattern used for link quality purposes in DP1.2. When NV_PDISP_SOR_DP_TPG is set to CSTM, the 80 bit pattern in these registers is send repetitively on the link. NV_PDISP_SOR_DP_TPG_LANE*_CHANNELCODING and NV_PDISP_SOR_DP_TPG_LANE*_SCRAMBLEREN should be set to _DISABLE in order to send out this pattern unaltered. For the HBR2 Compliance Eye Pattern, these three registers should all be set to 0, while NV_PDISP_SOR_DP_TPG_LANE*_CHANNELCODING and NV_PDISP_SOR_DP_TPG_LANE*_SCRAMBLEREN are set to _ENABLE.

- SOR_DP_LQ_CSTM0 contains bits 31:0
- SOR_DP_LQ_CSTM1 contains bits 63:32
- SOR_DP_LQ_CSTM2 contains bits 79:64

Offset: 0x6f | Byte Offset: 0x1bc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYM

24.6.86 SOR_NV_PDISP_SOR_DP_LQ_CSTM1_0

Offset: 0x70 | Byte Offset: 0x1c0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYM

24.6.87 SOR_NV_PDISP_SOR_DP_LQ_CSTM2_0

Offset: 0x71 | Byte Offset: 0x1c4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYM

24.7 DPAUX Registers

24.7.1 DPAUX_CTXSW_0

Context switch register

Should be common to all modules. Includes the current channel/class (which is writable by software) and the next channel/class (which the hardware sets when it receives a context switch).

Any context switch request triggers an interrupt to the host and causes the new channel/class to be stored in NEXT_CHANNEL/NEXT_CLASS (see vmod/chexample). Software sees that there is a context switch interrupt and does the necessary operations to make the module ready to receive traffic from the new context. It clears the context switch interrupt and writes CURR_CHANNEL/CLASS to the same value as NEXT_CHANNEL/CLASS, which causes a context switch acknowledge packet to be sent to the host. This completes the context switch and allows the host to continue sending data to the module.

Context switches can also be 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: 0x0 | Byte Offset: 0x0 | Read/Write: R/W | Reset: 0xXXXXf800 (0bxxxxxxxxxxxxxxxx11111x0000000000)

Bit	R/W	Reset	Description
31:28	RO	X	NEXT_CHANNEL: Next requested channel
25:16	RO	X	NEXT_CLASS: Next requested class
15:12	RW	0xf	CURR_CHANNEL: Current working channel, reset to 'invalid'
11	RW	0x1	AUTO_ACK: Automatically acknowledge any incoming context switch requests 0 = MANUAL 1 = AUTOACK
9:0	RW	0x0	CURR_CLASS: Current working class

24.7.2 DPAUX_INTR_EN_AUX_0

A DP SOR port can generate 4 types of interrupts:

- PLUG_EVENT: The HPD line has been high for at least DPAUX_HPD_CONFIG_0_PLUG_MIN_TIME microseconds
- UNPLUG_EVENT: The HPD line has gone low for at least DPAUX_HPD_CONFIG_0_UNPLUG_MIN_TIME microseconds
- IRQ_EVENT: After a DP device has been connected, it can generate an interrupt request. This will cause the DP hardware to read the Sink status and then notify the RM with an IRQ_EVENT interrupt.
- AUX_DONE: This interrupt will signify the completion of an AUX transaction.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x1..0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	DISABLED	AUX_DONE: 0 = DISABLED 1 = ENABLED
2	DISABLED	IRQ_EVENT: 0 = DISABLED 1 = ENABLED
1	DISABLED	UNPLUG_EVENT: 0 = DISABLED 1 = ENABLED
0	DISABLED	PLUG_EVENT: 0 = DISABLED 1 = ENABLED

24.7.3 DPAUX_INTR_AUX_0

The interrupt status generated by the DP SOR over AUX logic can be determined by reading this register. A pending interrupt can be cleared by writing 1 to the bit associated with that interrupt.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x5..0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	NOT_PENDING	AUX_DONE: 0 = NOT_PENDING 1 = PENDING
2	NOT_PENDING	IRQ_EVENT: 0 = NOT_PENDING 1 = PENDING
1	NOT_PENDING	UNPLUG_EVENT: 0 = NOT_PENDING 1 = PENDING
0	NOT_PENDING	PLUG_EVENT: 0 = NOT_PENDING 1 = PENDING

24.7.4 DPAUX_DP_AUXDATA_WRITE_W0_0

SOR_DP_AUXDATA_WRITE_W0 (REG_0)
SOR_DP_AUXDATA_WRITE_W1 (REG_1)
SOR_DP_AUXDATA_WRITE_W2 (REG_2)
SOR_DP_AUXDATA_WRITE_W3 (REG_3)

Data register array for DisplayPort writes.

All 4 registers are combined into a 16 bytes data buffer. This buffer contains data portion of an AUX native write, as specified in the DisplayPort specification. An AUX native reads maximum 16 bytes of data. The buffer is little endian.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x9..0xc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	REG

24.7.5 DPAUX_DP_AUXDATA_WRITE_W1_0

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0xd..0x10 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	REG

24.7.6 DPAUX_DP_AUXDATA_WRITE_W2_0

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x11..0x14 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	REG

24.7.7 DPAUX_DP_AUXDATA_WRITE_W3_0

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x15..0x18 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	REG

24.7.8 DPAUX_DP_AUXDATA_READ_W0_0

DP_AUXDATA_READ_W0 (REG_0)
DP_AUXDATA_READ_W1 (REG_1)
DP_AUXDATA_READ_W2 (REG_2)
DP_AUXDATA_READ_W3 (REG_3)

Data register array for DisplayPort reads.

All 4 registers are combined into a 16 bytes data buffer. This buffer contains data portion of an AUX native read, as specified in the DisplayPort specification. An AUX native reads maximum 16 bytes of data. The buffer is little endian.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x19..0x1c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	REG

24.7.9 DPAUX_DP_AUXDATA_READ_W1_0

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x1d..0x20 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	REG

24.7.10 DPAUX_DP_AUXDATA_READ_W2_0

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x21..0x24 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	REG

24.7.11 DPAUX_DP_AUXDATA_READ_W3_0

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x25..0x28 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	REG

24.7.12 DPAUX_DP_AUXADDR_0

DP_AUXADDR

Address register for DisplayPort AUX channel transaction.

This address register describes the 20-bit address that the transaction reads from/writes to the sink AUX register address space. Refer to the DisplayPort specification for more details.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x29..0x2c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	REG

24.7.13 DPAUX_DP_AUXCTL_0

DP_AUXCTL

Main control register for AUX transactions.

Refer to the DisplayPort specification for more information about AUX channel transactions.

The AUX bus can transmit data at 1Mbps. A single AUX transaction may take up to 648 microseconds to complete:

- 66 µs for precharge, SYNC, COMMAND, ADDRESS, and LENGTH
- 16*8 µs DATA
- 2 µs SYNC/STOP
- 400 µs maximum delay before reply begins
- 34 µs for precharge, SYNC
- 8 µs for REPLY_TYPE
- 8 µs LENGTH
- 2 µs SYNC/STOP

  ---

  648 microseconds

If an HPD_IRQ arrives just before an AUX request occurs, hardware will use the AUX channel to read the Link/Sink status, and then process the software request. This will incur an additional delay of up to 568 microseconds before the software request begins. This brings the total worst-case reply time to 1216 microseconds.

The AUXCTL_SEMA_REQUEST and AUXCTL_SEMA_GRANT bits are the semaphore for the DisplayPort AUX channel. Both VBIOS and RM need to use the AUX channel. The AUX semaphore is requested by writing to AUXCTL_SEMA_REQUEST. The client must check the value of AUXCTL_SEMA_GRANT to see if it was awarded the semaphore.

To obtain the semaphore:

For RM:

```
Write _RM to AUXCTL_SEMA_REQUEST
read AUXCTL_SEMA_GRANT
if (AUXCTL_SEMA_GRANT == _RM)
     RM grabbed the AUX channel successfully
else
     RM failed to control the AUX
```

For VBIOS:

```
Write _VBIOS to AUXCTL_SEMA_REQUEST
read AUXCTL_SEMA_GRANT;
if (AUXCTL_SEMA_GRANT == _VBIOS)
     VBIOS grabbed the AUX channel successfully;
else
     VBIOS failed to control the AUX;
```

To release the semaphore:

After the AUX transaction is finished, software must write _RELEASE to SEMA_REQUEST. This can be used to put the AUX channel back into a good state if it gets corrupted.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x2d..0x30 | Read/Write: R/W | Reset: 0x0X000000 (0b0xxxxxxxxx00xxx00000xxx000000000)

Bit	R/W	Reset	Description
31	RW	DEASSERT	RST: This can be used to put the AUX channel back into a good state if it gets corrupted. Set to ASSERT to force a localized reset of the AUX logic. Set to DEASSERT to allow the AUX engine to run 0 = DEASSERT 1 = ASSERT
25:24	RO	X	SEMA_GRANT: The client must check the value of AUXCTL_SEMA_GRANT to see if it was awarded the semaphore. SEMA_GRANT_NONE: AUX channel available SEMA_GRANT_RM: AUX channel obtained by RM SEMA_GRANT_VBIOS: AUX channel obtained by VBIOS SEMA_GRANT_PMU: AUX channel obtained by PMU 0 = NONE 1 = RM 2 = VBIOS 3 = PMU
21:20	RW	RELEASE	SEMA_REQUEST: The AUX semaphore is requested by writing to AUXCTL_SEMA_REQUEST. The process for RM or VBIOS to obtain the AUX channel is: Set to _RM if aux is obtained by RM. Set to _VBIOS if aux is obtained by VBIOS. Set to RELEASE to relinquish control of the AUX channel. 0 = RELEASE 1 = RM 2 = VBIOS 3 = PMU
16	RW	DONE	TRANSACTREQ: Software uses this bit to request an AUX channel transaction. After CMD, CMDLEN, and AUXDATA_WRITE have been set, this field should be set to _TRIGGER (1). After the hardware receives the reply, the hardware clears the TRANSACTREQ to _DONE (0).

Bit	R/W	Reset	Description
			If an unplug event is detected in the middle of a transaction, the current transaction is aborted, and the UNPLUG_EVENT bit will be set in DPAUX_INTR_4. Further transactions while unplugged will still be sent. If DPAUX_HYBRID_PADCTL_MODE is set to _I2C, or DPAUX_HYBRID_SPARE_PAD_PWR is set to _POWERDOWN, no transactions will be sent out. Any transactions in flight will complete. This prevents any interference with the I2C functionality of the pad. At the completion of an AUX transaction, the hardware will write the results into the DPAUX_DP_AUXSTAT register. The reply type should be checked as well as the error bits in this register to determine if the request was successful 0 = DONE 1 = PENDING
15:12	RW	I2CWR	CMD: AUX Command. Refer to Section 2.4, AUX channel syntax, in the DisplayPort specification. I2CWR: I2C write I2CRD: I2C read I2CREQWSTAT: I2C write status request I2CMOTWR: I2C write, Middle of transaction (MOT) I2CMOTRD: I2C read, MOT MOTREQWSTAT: I2C write status request, (MOT) AUXWR: AUX write AUXRD: AUX read 0 = I2CWR 1 = I2CRD 2 = I2CREQWSTAT 4 = MOTWR 5 = MOTRD 6 = MOTREQWSTAT 8 = AUXWR 9 = AUXRD
8	RW	NO	ADDRESS_ONLY: Some I2C-over-AUX operations require address-only transactions. This bit modifies any AUX transaction into an address only transaction. The DP specification only uses Address-only transactions for I2CRD and I2CWR (with or without MOT set). NO: Send the AUX transaction normally. YES: Send an Address-only AUX transaction. Only Command and Address will be sent. The CMDLEN field is ignored; no data will be written. Since no data is written, the ACK/NACK for this command will not contain an M value. The values in DPAUX_DP_AUXSTAT_REPLY_M and DPAUX_DP_AUXSTAT_RX_ERROR will not be accurate after an Address-only transaction. 0 = NO 1 = YES
7:0	RW	0x0	CMDLEN: These bits determine the number of data bytes an AUX channel transaction writes or reads. An AUX transaction can read or write a maximum 16 bytes. Note: This field should be set to N-1, where N is the number of bytes in the transaction.

24.7.14 DPAUX_DP_AUXSTAT_0

DP_AUXSTAT

When an AUX command completes, it will update most fields in this register.

Refer to the DisplayPort specification for more information about AUX channel transaction.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x31..0x34 | Read/Write: R/W | Reset: 0xX0XX00XX (0bxxxxxxxxxxxxxxxxxxxxx0000xxxxxxxx)

Bit	R/W	Reset	Description
28	RO	X	HPD_STATUS: Reports the current state of the HPD line, plugged or unplugged. This field is valid at all times, not just at the end of a transaction. This status is based on the value of the GPIO associated with this AUX channel. The connection between GPIO and AUX channel is hard coded, so they must be kept together when assigning them to an SOR link. AUX0 GPIO(1) AUX1 GPIO(19) AUX2 GPIO(15) AUX3 GPIO(21) 0 = UNPLUG 1 = PLUGGED
23:20	RO	X	AUXCTL_STATE: Reports the current state of the AUXCTL state machine. This field is valid at all times, not just at the end of a transaction. 0 = IDLE 1 = SYNC 2 = START1 3 = COMMAND 4 = ADDRESS 5 = LENGTH 6 = WRITE1 7 = READ1 8 = GET_M 9 = STOP1 10 = STOP2 11 = REPLY 12 = CLEANUP
19:16	RO	X	REPLYTYPE: These bits are for status and describe the reply type of the AUX channel command. Refer to Section 2.4.1 of the DisplayPort specification for more information about how to handle each type of reply. ACK: Transaction successful NACK: Transaction failed or is incomplete DEFER: Try again later. I2CNACK: I2C transaction failed or is incomplete I2CDEFER: Try again later. 0 = ACK 1 = NACK 2 = DEFER 4 = I2CNACK 8 = I2CDEFER
11	RW	NOT_PENDING	NO_STOP_ERROR: If the AUX reply from the Sink device did properly terminate with a STOP signal (if the AUX line goes idle), this error bit will be set. There may have been some problem with the connection. The results of the most recent request cannot be guaranteed correct. This bit will remain set until 1 is written to it. 0 = NOT_PENDING 1 = PENDING
10	RW	NOT_PENDING	SINKSTAT_ERROR: After an HPD IRQ, the hardware will automatically read DPCD addresses 0x205-0x200 and place the results in the DPAUX_DP_AUX_SINKSTAT registers. REPLYTYPE is not updated on this type of read, so if the Sink device does not respond with an ACK, this bit will be set. RX_ERROR, TIMEOUT, and NO_STOP can still be set by the automatic read. This bit will remain set until 1 is written to it. 0 = NOT_PENDING 1 = PENDING
9	RW	NOT_PENDING	RX_ERROR: This error reporting bit is set if there is an undefined error detected by the AUX channel. Currently, this is only set if the logic was expecting the sink to send a length value (M), but one was not received. This bit will remain set until 1 is written to it. 0 = NOT_PENDING 1 = PENDING
8	RW	NOT_PENDING	TIMEOUT_ERROR: This error reporting bit is set to PENDING when a timeout has occurred while waiting for a reply. The AUX channel will wait for 400 µs before aborting the AUX transaction. The timeout value is programmable in the

Bit	R/W	Reset	Description
			DPAUX_DP_AUX_CONFIG register. This bit will remain set until 1 is written to it. 0 = NOT_PENDING 1 = PENDING
7:0	RO	X	REPLY_M: These bits show the number of data bytes an AUX channel transaction receives. An AUX transaction can read or write 16 bytes maximum.

24.7.15 DPAUX_DP_AUX_SINKSTATLO_0

All bits of the SINKSTATLO and SINKSTATHI registers are combined into a 6-byte data buffer. This buffer contains sink/link status information from AUX address space from 00200h to 00205h, as specified in the DisplayPort specification. Hardware automatically issues an AUX read to the above address after a HPD IRQ is detected. The 6 bytes of read data are put into this buffer. The buffer is little endian.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x35..0x38 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	REG

24.7.16 P_AUX_SINKSTATHI_0

All bits of the SINKSTATLO and SINKSTATHI registers are combined into a 6-byte data buffer. This buffer contains sink/link status information from AUX address space from 00200h to 00205h, as specified in the DisplayPort specification. Hardware automatically issues an AUX read to the above address after a HPD IRQ is detected. The 6 bytes of read data are put into this buffer. The buffer is little endian.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x39..0x3c | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	REG

24.7.17 DPAUX_HPD_CONFIG_0

HPD_CONFIG

This register is used to configure the behavior of the HPD plug/unplug events. The INIT values are defined by the DisplayPort specification.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x3d..0x40 | Read/Write: R/W | Reset: 0x07d000fa (0b00000111110100000000000011111010)

Bit	Reset	Description
31:16	0x7d0	UNPLUG_MIN_TIME: The HPD line must be low for this long before it is considered an unplug event. The units of this field are in microseconds.
15:0	0xfa	PLUG_MIN_TIME: The HPD line must be high for this long to be considered a plug event. The units of this field are in microseconds.

24.7.18 DPAUX_HPD_IRQ_CONFIG_0

HPD_IRQ_CONFIG

This register controls the size of the HPD IRQ pulse. The INIT value is defined by the DisplayPort specification.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x41..0x44 | Read/Write: R/W | Reset: 0x000000fa (0bxxxxxxxxxxxxxxxx0000000011111010)

Bit	Reset	Description
15:0	0xfa	MIN_LOW_TIME: IRQ minimum time. The HPD line must be low for at least this amount of time to be an IRQ. An IRQ will occur if IRQ_MIN_TIME < duration of HPD low pulse < UNPLUG_MIN_TIME.

24.7.19 DPAUX_DP_AUX_CONFIG_0

AUX_CONFIG contains miscellaneous configuration parameters for the AUX channel.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x45..0x48 | Read/Write: R/W | Reset: 0x00000190 (0bxxxxxxxxxxxxxxxx0000000110010000)

Bit	Reset	Description
15:0	0x190	TIMEOUT: Sets the wait time, in microseconds, before an AUX transaction will abort and report TIMEOUT in DPAUX_DP_AUXSTAT_TIMEOUT. The default value of 400 µs is specified in the DisplayPort specification.

24.7.20 DPAUX_HYBRID_PADCTL_0

HYBRID_PADCTL

Configuration for the hybrid pads that can be used as AUX/I2C.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x49..0x4c | Read/Write: R/W | Reset: 0x00002462 (0bxxxxxxxxxxxxxxxx0010x10001100010)

Bit	Reset	Description
15	DISABLE	I2C_SDA_INPUT_RCV: Active high receiver enable for the I2C pad's DATA channel. 0 = DISABLE 1 = ENABLE
14	DISABLE	I2C_SCL_INPUT_RCV: Active high receiver enable for the I2C pad's CLK channel. 0 = DISABLE 1 = ENABLE
13:12	V0_70	AUX_CMH: Active high 1.0V signal to control the output common mode voltage. 0_60V : VCM = 0.6V (default) 0_64V : VCM = 0.64V 0_70V : VCM = 0.70V 0_56V : VCM = 0.56V 0 = V0_60 1 = V0_64 2 = V0_70 3 = V0_56
10:8	OHM_50	AUX_DRVZ: Active high driver output impedance control. Increasing this value decreases the driver impedance. 0 = OHM_78 1 = OHM_60 2 = OHM_54 3 = OHM_45 4 = OHM_50

Bit	Reset	Description
		5 = OHM_42 6 = OHM_39 7 = OHM_34
7:2	0x18	AUX_DRVI: Active high output driver current control. Increasing the register value increases drive current.
1	ENABLE	AUX_INPUT_RCV: Active high receiver enable for the AUX CH pad. 0 = DISABLE 1 = ENABLE
0	AUX	MODE: Controls whether the pad is in I2C or AUX mode. 0 = AUX 1 = I2C

24.7.21 DPAUX_HYBRID_SPARE_0

PAD_PWR: When a HotPlug is detected on a given DP port, this bit must be set to _POWERUP before any AUX transactions are done. When an Unplug event is detected, this can be written to _POWERDOWN. As long as a DP monitor is connected, the AUX pads should remain powered on because the monitor may send an HPD IRQ, which will require the HW to initiate an AUX read without SW intervention. The logic assumes that the AUX pad will be powered on.

Offset: 0x4d..0x50 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
0	POWERUP	PAD_PWR: Controls the E_PWRD port of the hybrid pad. The pads can consume a lot of power if enabled and should be disabled when not in use. POWERUP: pad has power and can be used for transactions. POWERDOWN: pad is powered down and cannot be used. 0 = POWERUP 1 = POWERDOWN

24.7.22 DPAUX_SCRATCH_REG0_0

SCRATCH_REG0

This register is used in the sor_wrap to control the BFM parameters. For fields such as data/address/cmd_type/data_count, the corresponding values get loaded from the scratch_reg1 registers

Offset: 0x51 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	REG: 0 = IDLE 1 = CLEAR_AUTO_RESP 2 = CLEAR_WR_DEFER 3 = CLEAR_WR_NAK 4 = CLEAR_WR_PARTIAL 5 = CLEAR_RD_DEFER 6 = CLEAR_RD_NAK 7 = CLEAR_RD_PARTIAL 8 = SET_AUTO_RESP 9 = SET_WR_DEFER 10 = SET_WR_NAK 11 = SET_WR_PARTIAL 12 = WR_DATA0 13 = WR_DATA1 14 = WR_DATA2 15 = WR_DATA3 16 = WR_DATA4 17 = WR_DATA5 18 = WR_DATA6

Bit	Reset	Description
		19 = WR_DATA7 20 = WR_DATA8 21 = WR_DATA9 22 = WR_DATA10 23 = WR_DATA11 24 = WR_DATA12 25 = WR_DATA13 26 = WR_DATA14 27 = WR_DATA15 28 = SET_RD_DEFER 29 = SET_RD_NAK 30 = SET_RD_PARTIAL 31 = RD_RESP_DATA0 32 = RD_RESP_DATA1 33 = RD_RESP_DATA2 34 = RD_RESP_DATA3 35 = RD_RESP_DATA4 36 = RD_RESP_DATA5 37 = RD_RESP_DATA6 38 = RD_RESP_DATA7 39 = RD_RESP_DATA8 40 = RD_RESP_DATA9 41 = RD_RESP_DATA10 42 = RD_RESP_DATA11 43 = RD_RESP_DATA12 44 = RD_RESP_DATA13 45 = RD_RESP_DATA14 46 = RD_RESP_DATA15 47 = RX_CMD 48 = RX_ADDR 49 = RX_DATA_CNT 50 = SET_RX_EXECUTE 51 = CLEAR_RX_EXECUTE 52 = SET_HPD_START 53 = CLEAR_HPD_START 54 = HPD_INIT_VALUE 55 = HPD_END_VALUE 56 = HPD_PULSE_WIDTH 57 = HPD_DISABLE 58 = RELEASE_ALL 59 = START_TIMER 60 = END_TIMER 61 = IF_REG_TEST 62 = SET_RESPONSE_TIMER 63 = RELEASE_RESPONSE_TIMER 64 = SET_RESPONSE_FAULTY 65 = CLEAR_RESPONSE_FAULTY 66 = START_WAIT_TIMER 67 = CLEAR_WAIT_TIMER 68 = SET_IGNORE_AUX_IN 69 = CLEAR_IGNORE_AUX_IN 70 = SET_AUX_IF_TIMER_RESET 71 = CLEAR_AUX_IF_TIMER_RESET 72 = SET_RD_PARTIAL_DATA_COUNT 73 = CLEAR_RD_PARTIAL_DATA_COUNT 74 = SET_WR_PARTIAL_DATA_COUNT 75 = CLEAR_WR_PARTIAL_DATA_COUNT 76 = WR_EXPECTED_MVID 77 = RD_MVID_CHECK 78 = RD_ATTACHED 79 = SET_PWM_START_MON 80 = CLEAR_PWM_START_MON 81 = READ_XTALCLK_COUNT 82 = READ_PCLK0_COUNT 83 = READ_PCLK1_COUNT 84 = READ_PCLK2_COUNT 85 = READ_MONITOR_FAILED 86 = SET_CLK_REC_PATTERN_CHK_DISABLED 87 = SET_FORCE_RESET

Bit	Reset	Description
		88 = SET_TARGET_CR_DRIVE_CURRENT 89 = SET_TARGET_EQ_PRE_EMPHASIS 90 = SET_TARGET_CR_BW 91 = SET_CR_UPDATE_US 92 = CLEAR_CLK_REC_PATTERN_CHK_DISABLED 93 = CLEAR_FORCE_RESET 94 = CLEAR_TARGET_CR_DRIVE_CURRENT 95 = CLEAR_TARGET_CR_BW 96 = CLEAR_CR_UPDATE_US 97 = SET_EQ_UPDATE_US 98 = SET_EQ_PATTERN_CHK_DISABLED 99 = CLEAR_EQ_PATTERN_CHK_DISABLED 100 = RD_AUX_BFM_TIMER 101 = RD_PATTERN_RCVD_STATUS 102 = CLEAR_LANE_PATTERN 103 = ENABLE_LANE_SEQ_MON 104 = DISABLE_LANE_SEQ_MON 105 = READ_PDTXD 106 = PWM_MON_CTL 107 = READ_PWM_COUNTER 108 = READ_PWM_STATUS

24.7.23 DPAUX_SCRATCH_REG1_0

SCRATCH_REG1

This register is used for loading data/address/cmd_type/data_count values when the corresponding field in the SCRATCH_REG0 is triggered.

Offset: 0x55 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	REG

24.7.24 DPAUX_SCRATCH_REG2_0

SCRATCH_REG2

This register is used to provide data back to the .tcl tests in cases where tests wait/poll for a particular value.

Offset: 0x59 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	REG: 0 = IDLE 1 = DONE

25.0 HDMI CEC

The HDMI Consumer Electronics Control (CEC) block supports CEC standard communication over an HDMI connection. It supports both remote control of HDMI devices attached to the Tegra® K1 device, and also allows other HDMI devices to control Tegra functions. Refer to the CEC appendix of the HDMI specification for details of this link.

25.1 Functional Description

The CEC module is an APB slave. It is located at address NV_ADDRESS_MAP_APB_CEC_BASE (0x70015000).

The CEC consists of hardware state machines that send and receive bytes over the CEC wire. Its simple host interface allows one byte to be sent and received at a time.

The CEC has interrupts for interrupt-driven input and output, and also permits polling I/O.

25.2 Programming Guidelines

25.2.1 Initialization

25.2.1.1 Clock and Reset

The CEC uses a fixed clock source – the APB bus clock. It also has a clock enable and reset in the CAR block.

- CAR.RST_DEVICES_W.SWR_CEC_RST = DISABLE
- CAR.CLK_OUT_ENB_W.CLK_ENB_CEC = ENABLE

25.2.1.2 Interrupt

The CEC is on PRI interrupt controller bit 3.

CEC INT_MASK should be enabled for TX_* and RX_* interrupts. TX_REGISTER_EMPTY and RX_REGISTER_FULL are the key interrupts; most others are errors.

Refer to the Interrupt Controller section for more information.

25.2.1.3 Pin Mux

The CEC is available on the HDMI_CEC pin. HDMI_CEC pin mux should be configured for:

- PM=0 (primary CEC function)
- PUPD=NORMAL
- TRISTATE=NORMAL
- E_INPUT=ENABLE
- OD=ENABLE.

25.2.2 CEC Timing

Timing parameters are in units of 32 microseconds. See the HDMI specification for CEC timing requirements. In Tegra K1 devices, the registers have default (reset) values as indicated in the following table.

Register	Field	Value	Comment
RX_TIMING_0	RX_START_BIT_MAX_LO_TIME	0x7a	

Register	Field	Value	Comment
	RX_START_BIT_MIN_LO_TIME	0x6d	
	RX_START_BIT_MAX_DURATION	0x93	
	RX_START_BIT_MIN_DURATION	0x86	
RX_TIMING_1	RX_DATA_BIT_MAX_LO_TIME	0x35	
	RX_DATA_BIT_SAMPLE_TIME	0x21	
	RX_DATA_BIT_MAX_DURATION	0x56	
	RX_DATA_BIT_MIN_DURATION	0x40	
RX_TIMING_2	RX_END_OF_BLOCK_TIME	0x50	
TX_TIMING_0	TX_START_BIT_LO_TIME	0x74	
	TX_START_BIT_DURATION	0x8d	
	TX_BUS_XITION_TIME	0x8	
	TX_BUS_ERROR_LO_TIME	0x71	
TX_TIMING_1	TX_LO_DATA_BIT_LO_TIME	0x2f	
	TX_HI_DATA_BIT_LO_TIME	0x13	
	TX_DATA_BIT_DURATION	0x4b	
	TX_ACK_NAK_BIT_SAMPLE_TIME	0x21	
TX_TIMING_2	BUS_IDLE_TIME_ADDITIONAL_FRAME	0x7	
	BUS_IDLE_TIME_NEW_FRAME	0x5	
	BUS_IDLE_TIME_RETRY_FRAME	0x3	

25.2.3 CEC Control

The CEC has several modes but only some are simulated or supported. TX_RX_MODE=ENABLE should be set last.

Register	Field	Value	Comment
SW_CONTROL	MODE	DISABLE	Use HW mode
INPUT_FILTER	FIFO_LENGTH	0	
	MODE	DISABLE	
HW_CONTROL	RX_LOGICAL_ADDRS	*Slave addresses (bitmap)*	If slave, holds addresses
	RX_SNOOP	DISABLE	
	RX_NAK_MODE	BLOCK	
	TX_NAK_MODE	BLOCK	
	FAST_SIM_MODE	DISABLE	SW should use DISABLE
	TX_RX_MODE	ENABLE	Enable last
INT_MASK	TX_*, RX_*	DISABLE/ENABLE	ENABLE to enable interrupt

25.2.4 Transmission

Each message is a series of bytes, and each byte has several flags associated with it.

Register	Field	Value	Comment
TX_REGISTER	DATA	*byte*	8-bit byte
	EOM	1 on last byte	End of message flag
	ADDRESS_MODE	DIRECT/BROADCAST	
	GENERATE_START_BIT	1	
	RETRY_FRAME	0/1	

In general, the first byte contains source and destination addresses, per the CEC specification. The source address is chosen by software as the Tegra address. The destination address is either 15 for broadcast, or less than 15 for a particular slave. The last byte contains EOM=1. When sending to address 15 (broadcast), ADDRESS_MODE must be BROADCAST so that ACK and NAK have correct polarity. If this transmission is a retry of a previous one, RETRY_FRAME should be 1 so that correct retry timing is used.

The TX register interface is unusual in that the interrupt (TX_REGISTER_EMPTY) controls transmission – it's not merely a status signal. To transmit a byte, TX_REGISTER is first written with data, then the INT_STAT.TX_REGISTER_EMPTY interrupt must be cleared. If interrupt service routines are used, they must be careful when clearing an interrupt, because doing so incorrectly could cause a spurious or lost byte.

Transmission pseudo-code:

```
Transmit(unsigned char data, bool eom?, bool broadcast?):
        // enqueue byte
        Write TX_REGISTER: DATA = data,
                EOM = eom?,
                ADDRESS_MODE = broadcast?,
                GENERATE_START_BIT = 1
        // clear interrupt to enable transmission
        Write INT_STAT: TX_REGISTER_EMPTY = 1
        // wait for transmission
        Either Poll INT_STAT.TX_* to go high, or wait for TX_* interrupt.
        If TX_REGISTER_EMPTY=1, transfer is complete.
        If other TX_* interrupts set, then error.
```

Transmission of multi-block frames requires the blocks (bytes) to be sent back-to-back with no idle space between. If software does not provide the bytes in time, hardware will signal a TX_REGISTER_UNDERRUN interrupt and halt transmission (see Error Recovery below).

25.2.5 Reception

The RX_REGISTER_FULL interrupt indicates that a byte has been received and blocks further reception until cleared. If a subsequent byte is received while RX_REGISTER_FULL is high, an RX_REGISTER_OVERRUN error results.

When a byte is received, two additional EOM and ACK flags are provided as well.

Register	Field	Value	Comment
RX_REGISTER	DATA	*byte*	8-bit byte
	EOM	1 on last byte	End of message flag
	ACK	*bit from bus*	Expected value depends on whether broadcast or direct

Once INT_STAT indicates a byte has been received, first read the byte (and flags), and then clear the interrupt.

Reception pseudo-code:

```
Receive( unsigned char &data, bool &eom, bool &ack):
        // wait for data
        Either Poll INT_STAT.RX_REGISTER_FULL, or
                wait for CEC RX_REGISTER_FULL interrupt
        If INT_STAT.RX_* error interrupts set, then error
        data, eom, ack = Read RX_REGISTER.DATA/EOM/ACK
        // clear interrupt to allow further reception
        Write INT_STAT.RX_REGISTER_FULL = 1
```

25.2.6 Error Recovery

In case of transmission error such as UNDERRUN or NAKD, the transmission will stop. Alternatively, the soft reset sequence should be used:

```
tmp = CEC_HW_CONTROL
CEC_HW_CONTROL = 0
CEC_INT_STATUS = 0xffffffff
CEC_HW_CONTROL = tmp
```

25.3 CEC Registers

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

For more information on Consumer Electronics Control (CEC), refer to the CEC appendix of the HDMI specification.

25.3.1 CEC_SW_CONTROL_0

SW controlled mode is not supported. Leave disabled.

Offset: 0x0 | Read/Write: R/W | Reset: 0x000000XX (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	DISABLE	MODE: 0 = DISABLE 1 = ENABLE
4	RO	X	FILTERED_RX_DATA_PIN
0	RO	X	RAW_INPUT_DATA_PIN

25.3.2 CEC_HW_CONTROL_0

The following steps are the best way to reset the CEC engine:

1. Set the CEC_HW_CONTROL_TX_RX_MODE to DISABLE
2. Set all the interrupt enable bits in CEC_HW_INTR_EN to DISABLE
3. Wait 1 second (1 second is the maximum CEC bus timeout period from the HDMI specification)
4. Set the CEC_HW_CONTROL_TX_RX_MODE to ENABLE and restart.

Other devices on the bus might have been communicating just fine. When the restart has completed, there might be a few spurious false start bits and other receive bus anomalies before normal operation is resumed.

The CEC_HW_CONTROL register contains the control bits and fields for operating and configuring the HW CEC engine.

Note:	Once the block is enabled, software should not attempt to change these configuration parameters without first disabling the block, except for the RX_LOGICL_ADDRS and RX_SNOOP fields.

RX_LOGICAL_ADDRS

The HDMI specification permits a single physical entity to claim logical addresses reserved for different functions (e.g., "Tuner x" and "Playback device y"). For a PC, it is entirely possible to be a multi-function device, so hardware might need to respond to 2 (or more) logical addresses (as many combinations as make sense). The next fields are used to configure the logical addresses that hardware will respond to. As per the specification, hardware will always respond to the broadcast address. This is a 15-bit field, where each bit position is associated with one of the logical addresses. A '1' in any bit position causes the hardware to respond to the associated logical address (i.e., capture data blocks/frames addressed to the address and properly ack/nak said blocks). To keep things simple, the mapping of bit position to logical address is direct, i.e., bit N maps to logical address N.

RX_SNOOP

When enabled, the hardware will intercept and forward to software all traffic on the CEC bus. It will continue to ack/nak only those addresses that are assigned to it.

RX_NAK_MODE

The HDMI specification is a bit unclear about what the initiator is supposed to do if an intermediate block of a frame is NAK'd, i.e., should the initiator cease transmitting immediately or complete the rest of the frame. The HDMI specification says that a follower may NAK a frame at any time, but the concern is that some initiator might not recover from such an early NAK'd frame. This bit will permit the receive state machine to operate in either mode. The choices are:

- BLOCK which means that a frame will be NAK'd as soon as a RX_REGISTER_OVERRUN is detected
- FRAME which means that the hardware will keep track of any RX_REGISTER_OVERRUNs that occur during frame reception, and NAK only during the last block.

TX_NAK_MODE

The HDMI specification is a bit unclear about what the initiator is supposed to do if an intermediate block of a frame is NAK'd, i.e., should the initiator cease transmitting immediately or complete the rest of the frame. This bit will permit the transmit state machine to operate in either mode. The choices are:

- BLOCK which means transmission will cease at the first NAK'd block
- FRAME which means that transmission will always continue to the end of the frame.

TX_RX_INTERRUPT_ROUTING

The engine generates a single composite interrupt output signal which can then be routed to either PMU or HOST. This bit controls that routing.

TX_RX_MODE

This bit enables or disables the operation of the HW CEC engine. If the TX_RX_MODE is changed to DISABLE, then the HW engine returns to the IDLE state irrespective of what it is doing and drives a 1 onto the CEC bus. If CEC_SW_CONTROL_MODE is ENABLED, the HW CEC engine operation is automatically disabled.

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxx0xxxxxxx00000000000000000)

Bit	Reset	Description
31	DISABLE	TX_RX_MODE: 0 = DISABLE 1 = ENABLE
30	DISABLE	FAST_SIM_MODE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
24	BLOCK	TX_NAK_MODE: 0 = BLOCK 1 = FRAME
16	BLOCK	RX_NAK_MODE: 0 = BLOCK 1 = FRAME
15	DISABLE	RX_SNOOP: 0 = DISABLE 1 = ENABLE
14:0	0x0	RX_LOGICAL_ADDRS: All logical addresses that should be matched. One bit per address.

25.3.3 CEC_INPUT_FILTER_0

The CEC_INPUT_FILTER register is used to configure the HW filtering that is required to deglitch the incoming receive data line. As data arrives into the chip, it is pushed into a 1 bit wide FIFO with a maximum of 64 entries deep. The actual used depth of the FIFO is set using the CEC_INPUT_FILTER_FIFO_LENGTH field. The used length is equal to CEC_INPUT_FILTER_FIFO_LENGTH+1. A datum is pushed into the FIFO once per microsecond tick. The FIFO bits should reset to '1' (the idle condition of the CEC bus).

The filtered output of the FIFO is computed roughly as follows:

```
filterMsk[63:0] = (1 << CEC_INPUT_FILTER_FIFO_LENGTH + 1) - 1;

if (reset || (CEC_INPUT_FILTER_MODE == CEC_INPUT_FILTER_MODE_DISABLE))
    fifo[63:0] = 0xffffffffffffffff;
else if (clock tick time)
    fifo[63:0] = (fifo[63:0] << 1) | rawInputDataPin;


if (reset || (CEC_INPUT_FILTER_MODE == CEC_INPUT_FILTER_MODE_DISABLE))
    filteredRxDataPin = 1;
else if ((clock tick time) && ((fifo & filterMsk) == filterMsk))
    filteredRxDataPin = 1;
else if ((clock tick time) && ((~fifo & filterMsk) == filterMsk))
    filteredRxDataPin = 0;
else
    filteredRxDataPin = filteredRxDataPin;
```

This logic should make sure that the filteredRxDataPin output only transitions to 0 or 1 when all the examined bits in the FIFO are also 0 or 1. Thus the filtered data line will not transition as long as there are glitches in the FIFO. The length of 64 provides a maximum operational length of 64 µs.

Every state transition on the filtered receive data line is reportable to software via the interrupt register described later.

Offset: 0x8 | Read/Write: R/W | Reset: 0x000000XX (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	DISABLE	MODE: 0 = DISABLE 1 = ENABLE
5:0	X	FIFO_LENGTH

25.3.4 CEC_SPARE_0

Spare register for future use.

Offset: 0xc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	SPARES

25.3.5 CEC_TX_REGISTER_0

CEC_HW_TX_REGISTER register

This register is used by software to provide the hardware with the actual data to be transmitted on the bus. Note that hardware will 'blindly' transmit what it is given. For example, it will not check to be sure that a proper legal initiator address has been provided, it will not check to be sure that the maximum frame length of 16 is not violated, etc. These higher level protocol checks are the domain of the SW. In order to facilitate easier software programming and smoother operation, hardware will make its own working copy of the fields in this register, thus quickly freeing up the register for software to write the next block. (Basically, it is a one-deep FIFO with fullness reported via the TX_REGISTER_EMPTY bit).

DATA

The actual 8 bits of address/data to be transmitted on the bus. The data is always transmitted MSB (bit 7) first.

EOM

This is the "end of message" bit for this block of data. This bit is set at the last block of the frame.

ADDRESS_MODE

This bit indicates to the hardware whether this particular block is directly addressed or broadcast addressed (which in turn dictates how hardware is to interpret the ACK/NAK for the block).

GENERATE_START_BIT

Indicates to the hardware that it should precede the transmission of this block of data with a start bit.

RETRY_FRAME

Indicates if the current frame is a retry frame or not. Based on this value, hardware chooses an appropriate bus idle time programmed in TX_TIMING2 register and waits for the bus to be idle before it can attempt to send a start bit. This bit is only meaningful when TX_GENERATE_START_BIT is set.

RETRY FRAME	LAST_FRAM_ SENT_BY_US	WAIT TIME
1	YES	TIMING2_BUS_IDLE_TIME_RETRY_FRAME
1	NO	TIMING2_BUS_IDLE_TIME_RETRY_FRAME
0	NO	TIMING2_BUS_IDLE_TIME_NEW_FRAME
0	YES	TIMING2_BUS_IDLE_TIME_ADDITIONAL_FRAME

If a frame is being sent immediately following the previous frame, hardware waits for "TIMING2_BUS_IDLE_TIME_ADDITIONAL_FRAME". However, if someone else uses the bus before that time elapses, the hardware resets its wait time counter and waits for "TIMING2_BUS_IDLE_TIME_NEW_FRAME".

Offset: 0x10 | Read/Write: R/W | Reset: 0x00010000 (0bxxxxxxxxxxxxxxx01xxx0xxx000000000)

Bit	Reset	Description
17	DISABLE	RETRY_FRAME: 0 = DISABLE 1 = ENABLE
16	ENABLE	GENERATE_START_BIT: 0 = DISABLE 1 = ENABLE
12	DIRECT	ADDRESS_MODE: 0 = DIRECT 1 = BROADCAST
8	0x0	EOM
7:0	0x0	DATA

25.3.6 CEC_RX_REGISTER_0

CEC_HW_RX_REGISTER register

This register is used by hardware to buffer data to the software. When a block of data has been received from the bus, it is stored here for software (and the RX_REGISTER_FULL bit is set). The hardware also has a 'working copy' of the receive register to give software as much time as possible to empty it. When a full block is assembled, hardware places it into RX_REGISTER, so the register is like a 1 deep FIFO.

DATA

The 8 bits of data that were read from the bus.

EOM

The EOM bit read from the bus.

ACK_NAK

The ACK/NAK bit as read from the bus. In some cases (broadcast block) even though the device may have ACK'd the frame, some other device on the bus may NAK the frame, and software would need to know this.

Offset: 0x14 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	ACK
8	X	EOM
7:0	X	DATA

25.3.7 CEC_RX_TIMING_0_0

CEC_HW_RX_TIMING0 register

Note: All timing registers must be configured before the HW CEC engine is enabled.

RX_START_BIT_MAX_LO_TIME

nominal 3.9 ms, (3900/32 = 122 intervals) = 3.904 ms

The maximum time the received bus can remain low, and still be considered the beginning of a proper start bit. If the start bit low time overruns this time, the start bit is considered invalid and is ignored.

RX_START_BIT_MIN_LO_TIME

nominal 3.5 ms, (3500/32 = 109 intervals) = 3.488 ms

The minimum time the received bus can remain low and still be considered the beginning of a proper start bit. If the start bit low time underruns this time, the start bit is considered invalid and is ignored.

RX_START_BIT_MAX_DURATION

nominal 4.7 ms, (4700/32 = 147 intervals) = 4.704 ms

The maximum total duration of the start bit from the first detected high-to-low transition to the high-to-low transition that begins the first address bit. If the start bit duration overruns this time, the start bit is considered invalid and is ignored.

RX_START_BIT_MIN_DURATION

nominal 4.3 ms (4300/32 = 134 intervals) = 4.288 ms

The minimum total duration of the start bit from the first detected high-to-low transition to the high-to-low transition that begins the first address bit. If the start bit duration underruns this time, the start bit is considered invalid and is ignored.

Offset: 0x18 | Read/Write: R/W | Reset: 0x86936d7a (0b10000110100100110110110101111010)

Bit	Reset	Description
31:24	0x86	RX_START_BIT_MIN_DURATION
23:16	0x93	RX_START_BIT_MAX_DURATION
15:8	0x6d	RX_START_BIT_MIN_LO_TIME
7:0	0x7a	RX_START_BIT_MAX_LO_TIME

25.3.8 CEC_RX_TIMING_1_0

CEC_HW_RX_TIMING1 register

Note: All timing registers must be configured before the HW CEC engine is enabled.

RX_DATA_BIT_MAX_LO_TIME

nominal 1.7 ms, (1700/32 = 53 intervals) = 1.696 ms

The maximum time the received bus can remain low and still be considered the beginning of a proper start bit. If the start bit low time overruns this time, the start bit is considered invalid and is ignored.

RX_DATA_BIT_SAMPLE_TIME

nominal 1.05 ms, (1050/32 = 33 intervals) = 1.056 ms

The time to sample that data bit.

RX_DATA_BIT_MAX_DURATION

nominal 2.75 ms, (2750/32 = 86 intervals) = 2.752 ms

The maximum total duration of the start bit from the first detected high-to-low transition to the high-to-low transition that begins the first address bit. If the start bit duration overruns this time, the start bit is considered invalid and is ignored.

RX_DATA_BIT_MIN_DURATION

nominal 2.05 ms (2050/32 = 64 intervals) = 2.048 ms

The minimum total duration of the start bit from the first detected high-to-low transition to the high-to-low transition that begins the first address bit. If the start bit duration underruns this time, the start bit is considered invalid and is ignored.

Offset: 0x1c | Read/Write: R/W | Reset: 0x40562135 (0b01000000010101100010000100110101)

Bit	Reset	Description
31:24	0x40	RX_DATA_BIT_MIN_DURATION

Bit	Reset	Description
23:16	0x56	RX_DATA_BIT_MAX_DURATION
15:8	0x21	RX_DATA_BIT_SAMPLE_TIME
7:0	0x35	RX_DATA_BIT_MAX_LO_TIME

25.3.9 CEC_RX_TIMING_2_0

CEC_HW_RX_TIMING2 register

Note: All timing registers must be configured before the HW CEC engine is enabled.

RX_END_OF_BLOCK_TIME

nominal 1.9 ms, (1900/32 = 80 intervals) = 1.9 ms

The time to wait after the start of the final ACK/NAK phase of frame transmission before returning to the idle state. (Needed since the last ACK/NAK bit will not be followed by another high-to-low transition.)

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000050 (0bxxxxxxxxxxxxxxxxxxxxxxxx01010000)

Bit	Reset	Description
7:0	0x50	RX_END_OF_BLOCK_TIME

25.3.10 CEC_TX_TIMING_0_0

The next set of timing registers configures the CEC logic for HW assisted operation. To permit flexible configuration (to support possible semi-compliant devices), the bit timing control and check values are programmable (rather than just directly using the specification values).

While the block itself works on a 1 µs tick, in order to save register space, these timing numbers are specified in units of 32 µs.

First, the registers for the various timing intervals (there are quite a few of them).

CEC_HW_TX_TIMING0 register

Note: All timing registers must be configured before the HW CEC engine is enabled.

For details of exact values and tolerances, refer to the CEC appendix of the HDMI specification.

TX_START_BIT_LO_TIME

nominal 3.7 ms, (3700/32 = 116 intervals) = 3.712 ms

How long to hold the bus lo during the start bit.

TX_START_BIT_DURATION

nominal 4.5 ms, (4500/32 = 141 intervals) = 4.512 ms

The total duration of the start bit

TX_BUS_XITION_TIME

nominal 250 µs, (250/32 = 8 intervals) = 256 µs

Time to wait for bus to settle after transitioning output.

TX_BUS_ERROR_LO_TIME

nominal 3.6 ms (3600/32 = 113 intervals) = 3.616 ms

Time to drive bus low when bus error must be signaled on the bus

Offset: 0x24 | Read/Write: R/W | Reset: 0x71088d74 (0b01110001xxxx10001000110101110100)

Bit	Reset	Description
31:24	0x71	TX_BUS_ERROR_LO_TIME
19:16	0x8	TX_BUS_XITION_TIME
15:8	0x8d	TX_START_BIT_DURATION
7:0	0x74	TX_START_BIT_LO_TIME

25.3.11 CEC_TX_TIMING_1_0

CEC_HW_TX_TIMING1 register

Note: All timing registers must be configured before the HW CEC engine is enabled.

TX_LO_DATA_BIT_LO_TIME

nominal 1.5 ms, (1500/32 = 47 intervals) = 1.504 ms

How long to hold the bus low when transmitting a '0'.

TX_HI_DATA_BIT_LO_TIME

nominal 0.6 ms, (600/32 = 19 intervals) = 6.08 ms

How long to hold the bus low when transmitting a '1'.

TX_DATA_BIT_DURATION

nominal 2.4 ms, (2400/32 = 75 intervals) = 2.4 ms

The total duration of a data or ACK/NAK bit.

TX_ACK_NAK_BIT_SAMPLE_TIME

nominal 1.05 ms, (1050/32 = 33 intervals) = 1.056 ms

The time to sample that ACK/NAK bit.

Offset: 0x28 | Read/Write: R/W | Reset: 0x214b132f (0b00100001010010110001001100101111)

Bit	Reset	Description
31:24	0x21	TX_ACK_NAK_BIT_SAMPLE_TIME
23:16	0x4b	TX_DATA_BIT_DURATION
15:8	0x13	TX_HI_DATA_BIT_LO_TIME
7:0	0x2f	TX_LO_DATA_BIT_LO_TIME

25.3.12 CEC_TX_TIMING_2_0

CEC_HW_TX_TIMING2 register

This register contains the bus idle time values for the various scenarios indicated in the HDMI specification (section 9.1). Software indicates to the hardware the amount of delay in units of data bit periods, as defined by the CEC_HW_TX_TIMING1_TX_DATA_BIT_DURATION field.

ADDITIONAL_FRAME:

The amount of time the bus needs to be idle before a new frame can be sent immediately after sending a frame. The suggested value is greater than or equal to 7.

NEW_FRAME:

The amount of time the bus needs to be idle before a new frame can be sent if we were not the initiator for the previous frame. The suggested value is greater than or equal to 5.

RETRY_FRAME:

The amount of time the bus needs to be idle before the same frame can be resent. The suggested value is >= 3.

Offset: 0x2c | Read/Write: R/W | Reset: 0x00000357 (0bxxxxxxxxxxxxxxxxxxxxx001101010111)

Bit	Reset	Description
11:8	0x3	BUS_IDLE_TIME_RETRY_FRAME
7:4	0x5	BUS_IDLE_TIME_NEW_FRAME
3:0	0x7	BUS_IDLE_TIME_ADDITIONAL_FRAME

25.3.13 CEC_INT_STAT_0

CEC_HW_INTR register

This register contains the individual interrupt and status bits for the CEC HW unit. The CEC_HW_INTR_EN register contains the corresponding enable bit for each interrupt/status bit in the interrupt register. The interrupt enable determines whether the interrupt propagates to the PMIC as an interrupt, but the interrupt status bit itself is always set if the described condition occurs. Software writes a '1' to any interrupt/status bit in order to clear it.

Because the HW state machine also looks at the state of the interrupt bits when determining what action(s) to take, it is important for software to operate in roughly the following order when processing interrupts from this source:

1. Read the interrupt register to determine what caused the interrupt and what needs to be done.
2. Perform the operations required, e.g., reload TX register, read RX register.
3. Finally, clear the corresponding interrupt bits.

TX_REGISTER_EMPTY

The transmit register is empty, so software is free to write the next block of the frame into the register. For multi-block frames, software must in fact provide the next block before the current block is sent, otherwise a TX_REGISTER_UNDERRUN error will occur. When the TX engines reads from the TX_REGISTER register and stores a local copy, this interrupt is asserted.

TX_REGISTER_UNDERRUN

The transmit register was empty at the time the transmit hardware needed to have the next block ready to send. This is an error condition. The transmitter will have stopped transmitting, and recovery will require resending the entire frame from scratch.

TX_FRAME_OR_BLOCK_NAKD

Hardware sets this bit if any block/frame is NAK'd. If software sees this bit set, it knows it will have to resend the frame later.

TX_ARBITRATION_FAILED

Hardware sets this bit during the address block phase of transmitting a frame, if failed to win arbitration. In this case, transmission will cease, and the HW will flush any pending data in the TX_REGISTER. Software will need to retry the frame from scratch (after the appropriate signal free time, i.e., hardware will not automatically retry).

TX_BUS_ANOMALY_DETECTED

Sometime during block transmission, hardware detected anomalous behavior on the bus (e.g., the bus remained low after transmitter had signaled high). When such an anomaly is detected, the transmitter ceases operation, releases the bus high, and flushes any pending data in the TX_TRANSMIT register.

TX_FRAME_TRANSMITTED

This is asserted at the end of the last data bit, i.e., the ACK bit of the last block is sent. This is just an indication that the last block is sent but does not necessarily mean the transmission was successful. Software should still look for other interrupts to check for any errors.

RX_REGISTER_FULL

The hardware has assembled a complete block from the bus and placed it into the RX_REGISTER. Software should read the block immediately to ensure the RX_REGISTER is empty before hardware has the next block ready to load. When the HW RX engine writes RX data to the RX_REGISTER, this bit is set.

RX_REGISTER_OVERRUN

Software failed to read the RX_REGISTER before hardware had another block of data ready to transfer. Hardware will NAK the block/frame, and the initiator will need to retry later. Software will need to flush the partially accumulated frame.

RX_START_BIT_DETECTED

Whenever the receive engine detects a start bit, it will set this bit. This bit is used for debug.

RX_BUS_ANOMALY_DETECTED

The receiver detected some anomaly (including bus errors) on the bus. A bus error has a specific definition in the HDMI specification, but there are other kinds of anomalies that can occur. All anomalies and bus errors are reported here.

RX_BUS_ERROR_DETECTED

The receiver has detected the specific anomaly called a bus error and then signaled a bus error on the bus per the HDMI specification.

FILTERED_RX_DATA_PIN_TRANSITION_H2L

In direct SW bus drive mode, software needs to be interrupted whenever the state of the received data line transitions. This interrupt is asserted on a 1 to 0 transition.

FILTERED_RX_DATA_PIN_TRANSITION_L2H

In direct SW bus drive mode, software needs to be interrupted whenever the state of the received data line transitions. This interrupt is asserted on a 0 to 1 transition

Offset: 0x30 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14	X	FILTERED_RX_DATA_PIN_TRANSITION_L2H: 0 = INACTIVE 1 = ACTIVE
13	X	FILTERED_RX_DATA_PIN_TRANSITION_H2L: 0 = INACTIVE 1 = ACTIVE
12	X	RX_BUS_ERROR_DETECTED: 0 = INACTIVE 1 = ACTIVE
11	X	RX_BUS_ANOMALY_DETECTED: 0 = INACTIVE 1 = ACTIVE
10	X	RX_START_BIT_DETECTED: 0 = INACTIVE 1 = ACTIVE
9	X	RX_REGISTER_OVERRUN: 0 = INACTIVE 1 = ACTIVE
8	X	RX_REGISTER_FULL:

Bit	Reset	Description
		0 = INACTIVE 1 = ACTIVE
5	X	TX_FRAME_TRANSMITTED: 0 = INACTIVE 1 = ACTIVE
4	X	TX_BUS_ANOMALY_DETECTED: 0 = INACTIVE 1 = ACTIVE
3	X	TX_ARBITRATION_FAILED: 0 = INACTIVE 1 = ACTIVE
2	X	TX_FRAME_OR_BLOCK_NAKD: 0 = INACTIVE 1 = ACTIVE
1	X	TX_REGISTER_UNDERRUN: 0 = INACTIVE 1 = ACTIVE
0	X	TX_REGISTER_EMPTY: 0 = INACTIVE 1 = ACTIVE

25.3.14 CEC_INT_MASK_0

Mask register for interrupts. When a field in this register is set to ENABLE, the corresponding field with a value of 1 in the INT_STATUS register will result in assertion of an interrupt line.

Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx0000000xx000000)

Bit	Reset	Description
14	DISABLE	FILTERED_RX_DATA_PIN_TRANSITION_L2H: 0 = DISABLE 1 = ENABLE
13	DISABLE	FILTERED_RX_DATA_PIN_TRANSITION_H2L: 0 = DISABLE 1 = ENABLE
12	DISABLE	RX_BUS_ERROR_DETECTED: 0 = DISABLE 1 = ENABLE
11	DISABLE	RX_BUS_ANOMALY_DETECTED: 0 = DISABLE 1 = ENABLE
10	DISABLE	RX_START_BIT_DETECTED: 0 = DISABLE 1 = ENABLE
9	DISABLE	RX_REGISTER_OVERRUN: 0 = DISABLE 1 = ENABLE
8	DISABLE	RX_REGISTER_FULL: 0 = DISABLE 1 = ENABLE
5	DISABLE	TX_FRAME_TRANSMITTED: 0 = DISABLE 1 = ENABLE
4	DISABLE	TX_BUS_ANOMALY_DETECTED: 0 = DISABLE 1 = ENABLE
3	DISABLE	TX_ARBITRATION_FAILED: 0 = DISABLE 1 = ENABLE
2	DISABLE	TX_FRAME_OR_BLOCK_NAKD: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	DISABLE	TX_REGISTER_UNDERRUN: 0 = DISABLE 1 = ENABLE
0	DISABLE	TX_REGISTER_EMPTY: 0 = DISABLE 1 = ENABLE

25.3.15 CEC_HW_DEBUG_RX_0

Offset: 0x38 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
27	X	RXDATABIT_SAMPLE_TIMER
26	X	LOGICADDR_MATCH
25	X	FORCELOOUT
24:21	X	STATE
20:17	X	RXBIT_COUNT
16:0	X	DURATION_COUNT

25.3.16 CEC_HW_DEBUG_TX_0

Offset: 0x3c | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
26	X	TXDATABIT_SAMPLE_TIMER
25	X	FORCELOOUT
24:21	X	STATE
20:17	X	TXBIT_COUNT
16:0	X	DURATION_COUNT

25.3.17 CEC_HW_SPARE_0_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH

[THIS PAGE INTENTIONALLY LEFT BLANK]

26.0 MIPI-CSI (CAMERA SERIAL INTERFACE)

This section describes the Camera Serial Interface (CSI), which is based on the MIPI CSI 2.0 standard specification. The MIPI CSI 2.0 standard is available from MIPI to its members at http://www.mipi.org/.

26.1 Functional Description

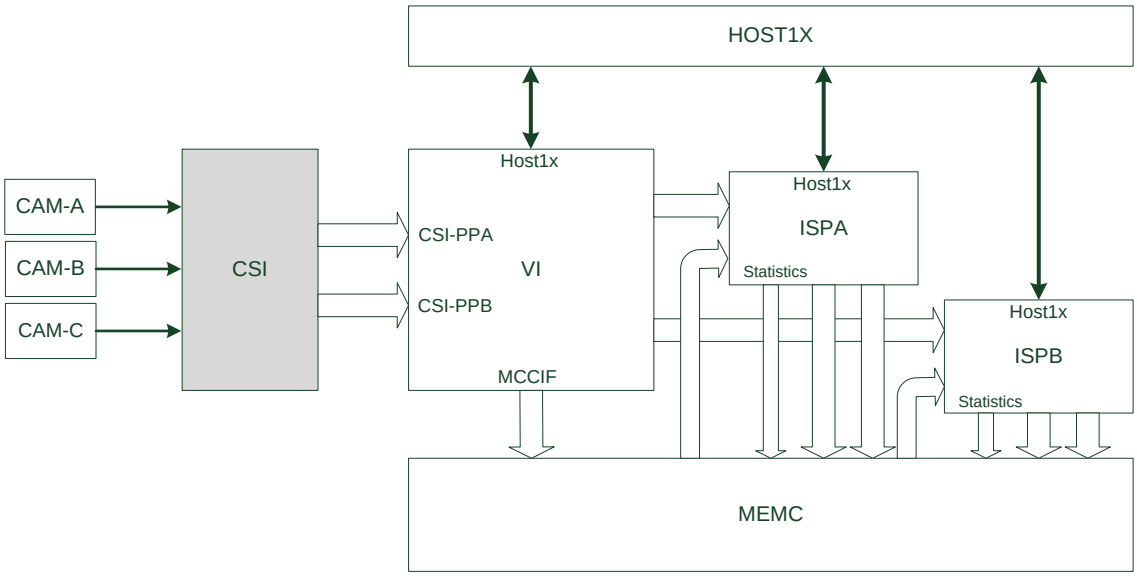
The CSI implements a MIPI compliant CSI receiver that may receive data from an external camera module with a CSI transmitter. The CSI is designed to provide for direct connection to three camera modules and allow any two of the modules to be active simultaneously:

- CSI-A interface supports 1 clock lane and up to 4 data lanes for Camera A.
- CSI-B interface supports 1 clock lane and up to 4 data lanes for Camera B OR a single data lane for Camera C.

The CSI offers both single and multi-camera configuration options. These options include a single camera configuration with a 1 to 4 lane sensor connected to CSI-A or CSI-B or a dual camera configuration with a 1 to 4 lane sensor connected to CSI-A and a 1 lane sensor to CSI-B.

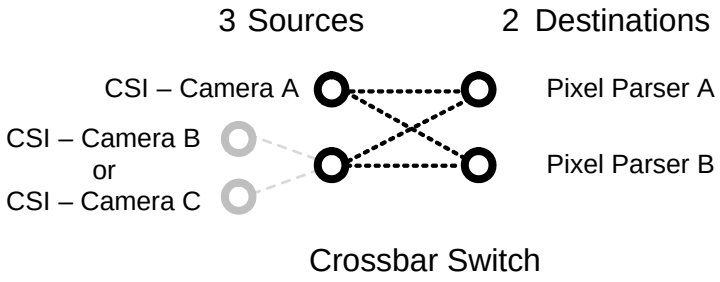
The following figure illustrates the datapaths associated with the Tegra® K1 camera subsystem.

Figure 84: Datapath Through the Tegra K1 Camera Subsystem



A maximum of two data/pixel streams can be processed simultaneously at any given time. The two streams can come from any one or two of the three possible sources, as shown in Figure 85. If the two streams come from a single source, then the streams are separated using a filter indexed on different virtual channel numbers or data types. In case of separation using data types, the normal data type is separated from the embedded data type. Because there are only two pixel parsers (PPA and PPB), the virtual channel and embedded data capability cannot be used at the same time.

Figure 85: Data Source/Destination Options



The CSI may receive data from camera modules through the MIPI serial interface. A test pattern generator has been implemented to assist validation.

The CSI supports single-shot mode and burst capture mode.

The CSI is designed with error resilience.

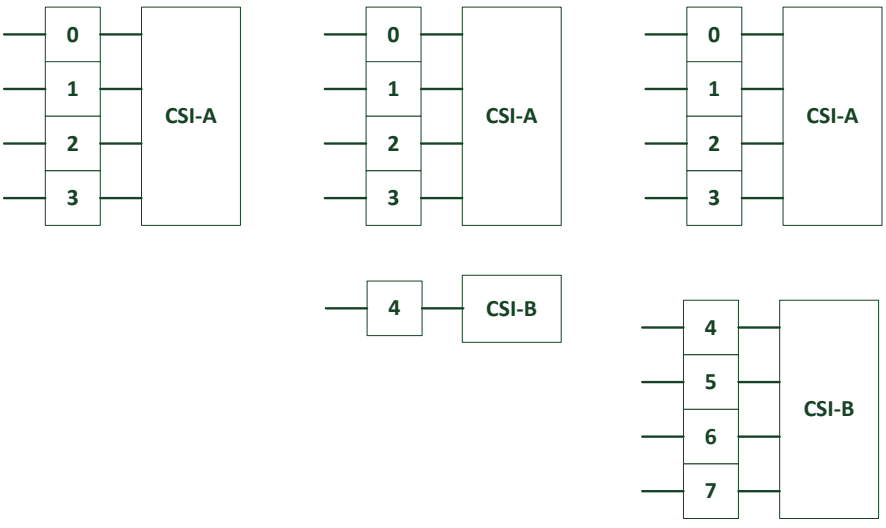
26.1.1 Lane Configurations

The number of MIPI lanes supported provides flexibility in the support of both single and dual camera user models. The typical single and dual camera application targets include high-resolution mono image capture in addition to dual camera image capture configurations:

- 1 x4 (Single camera with 4 lane sensor)
- 1 x4 + 1 x1 (one high resolution camera and another front facing low resolution camera)
- 2 x4 (Dual cameras for stereo with up to 4 lanes for each camera)

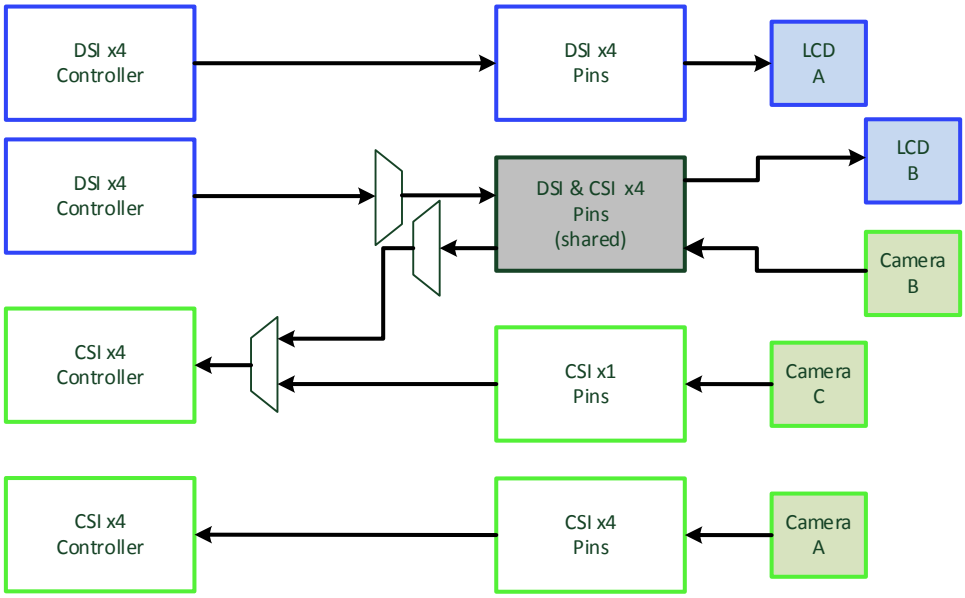
The following figure summarizes the typical camera configurations.

Figure 86: CSI MIPI Lane Configurations



The DSI and CSI controllers share one of the x4 MIPI bricks as shown resulting in several possible configuration options for the CSI depending on the system configuration. In some cases, the DSI x4 brick may not be available and only the CSI dedicated x4 brick will be available for a high-resolution camera interface.

Figure 87: Pin Muxing for DSI and CSI Controllers



26.2 Use Cases

The primary mono and stereo camera user models involve the capture of either pre-processed image data from a camera module in YCbCr 4:2:2 format or as a Raw Bayer stream with 10-bit precision. Additional pixel formats for image data may be accepted; however Raw 10-bit and YCbCr 4:2:2 are used primarily. Depending on the mode of operation, image resolution, and resulting pixel rate, one of several paths through the VI may be used. The use cases supported by the CSI include basic single camera, multiple cameras, embedded data, and virtual channel. The still and video capture sequences involve different single camera and multi-camera configuration options.

Table 103 summarizes the use case categories.

Table 103: Summary of CSI Camera Use Cases

Case	Description	Sources	Destinations
Single Camera Video Capture	Basic preview or video capture using single camera	Camera A or B or C	PPA or PPB
Stereo Camera Video Capture	Stereo pair video capture	Camera A and B	PPA and PPB
Single Camera Still Capture and Preview	Basic high resolution still capture supporting single shot and burst capture and preview	Camera A or B or C	PPA or PPB
Stereo Camera Still Capture and Preview	Stereo camera high resolution still capture and preview	Camera A and B	PPA and PPB
Dual Camera Mono Video Capture	Video feed of front facing camera and user facing camera for video annotation or video preview during Video Conference	Camera A and (B or C)	PPA and PPB
Video Conference Mono Still Capture & Preview*	Video feed of user facing camera for video conference and mono still capture from front facing camera.	Camera A and (B or C)	PPA and PPB
Dual Camera Automotive	Low resolution YUV or RGB video feed from two cameras.	Camera A and (B or C)	PPA and PPB

Case	Description	Sources	Destinations
Virtual Channel	One stream goes to both parsers. Packet filtering based on VC number in header	Camera A and B	PPA and/or PPB
Embedded Data	Mode 1: Each stream goes to one parser and has normal and embedded data	Camera A and/or (B or C)	PPA and PPB
	Mode 2: Embedded data from one stream goes to one pixel parser and the normal data goes to the other.	Camera A or (B or C)	PPA and PPB

The dual camera use cases require two capture streams from two different sensors, each imaging a separate scene. Within the Tegra K1 camera system, it is necessary to process two different images with different algorithm sequences/ISP setup or to receive a preprocessed YCbCr image and deliver the stream to system memory.

26.3 Performance

The following table describes the various MIPI lane configurations and the theoretical maximum Mpixels/s achievable at 100% efficiency for several RAW Bayer pixel precisions and YCbCr 4:2:2.

Table 104: Maximum Capture Rate in MPixels/s for Various CSI Configurations

Link Speed	Number of Lanes	Maximum Mpixels/s with 100% Efficiency				
		RAW8	RAW10	RAW12	RAW14	YCbCr422
600 Mbps	1	75	60	50	42.9	37.5
	2	150	120	100	85.7	75
	3	225	180	150	128.6	112.5
	4	300	240	200	171.4	150
800 Mbps	1	100	80	66.7	57.1	50
	2	200	160	133.3	114.3	100
	3	300	240	200	171.4	150
	4	400	320	266.7	228.6	200
1 Gbps	1	125	100	83.3	71.4	62.5
	2	250	200	166.7	142.9	125
	3	375	300	250	214.3	187.5
	4	500	400	333.3	285.7	250
1.5 Gbps	1	187.5	150	125	107.1	93.8
	2	375	300	250	214.3	187.5
	3	562.5	450	375	321.4	281.3
	4	750	600	500	428.6	375

The following table translates the maximum pixel rate into the maximum achievable frame rate for different sensor resolutions and bit depths.

Table 105: Maximum Capture Rate in FPS for Various CSI Configurations

Link Speed	Pixel Type	Maximum Frame Capture Rate (FPS) with 100% Efficiency											
		800 Mbps				1 Gbps				1.5 Gbps			
		1	2	3	4	1	2	3	4	1	2	3	4
8MP	RAW10	10	20	30	40	12.5	25	37.5	50	18.8	37.6	56.4	75.2
	RAW12	8.3	16.7	25	33.4	10.4	20.8	31.1	41.6	15.6	31.2	46.8	62.4
	RAW14	7.2	14.3	21.5	28.6	8.9	17.9	26.8	35.7	13.4	26.8	40.2	53.6
14MP	RAW10	5.5	10.9	16.4	21.9	6.8	13.7	20.5	27.3	10.7	21.4	32.1	42.8
	RAW12	4.6	9.1	13.7	18.2	5.7	11.4	17.1	22.8	8.9	17.8	26.7	35.6
	RAW14	3.9	7.8	11.7	15.6	4.9	9.8	14.7	19.5	7.7	15.4	23.1	30.8
16MP	RAW10	4.9	9.9	14.8	19.8	6.2	12.4	18.6	24.7	9.4	18.8	28.2	37.6
	RAW12	4.1	8.2	12.4	16.5	5.2	10.3	15.5	20.6	7.8	15.6	23.4	31.2
	RAW14	3.5	7.1	10.6	14.1	4.4	8.8	13.3	17.7	6.7	13.4	20.1	26.8
24MP	RAW10	3.3	6.6	9.8	13.1	4.1	8.2	12.3	16.4	6.3	12.6	18.9	25.2
	RAW12	2.7	5.5	8.2	10.9	3.41	6.8	10.3	13.7	5.2	10.4	15.6	20.8
	RAW14	2.3	4.7	7	9.4	2.9	5.9	8.8	11.7	4.5	9	13.5	18
32MP	RAW10	2.5	5.1	7.6	10.2	3.2	6.4	9.5	12.7	4.7	9.4	14.1	18.8
	RAW12	2.1	4.2	6.4	8.5	2.7	5.3	8	10.6	3.9	7.8	11.7	15.6
	RAW14	1.8	3.6	5.5	7.3	2.3	4.5	6.8	9.1	3.3	6.6	9.9	13.2

To minimize the rolling shutter artifacts a generally accepted minimum still capture target of 1/15th of a second is desired. In many cases, this frame rate can be managed since the high resolution CMOS sensor modules are unlikely to deliver the full resolution image at 15fps maximum. The Zero Shutter Lag (ZSL) user model in which a preview stream and a still capture circular buffer are maintained makes it desirable to achieve the highest frame rate possible to provide good motion video for scene composition.

26.4 Supported CSI to VI Data Formats

The formats of the pixel data presented from the CSI to the VI across the 24-bit bus for each of the pixel parsers are summarized in Table 106.

The Tegra K1 device presents all Raw data to the VI/ISP in its original bit order.

Table 106: CSI to VI Pixel Formats

Format Name	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
6-bit raw or DPCM 12-6-12/10-6-10	0[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
7-bit raw or DPCM 12-7-12/10-7-10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D6	D5	D4	D3	D2	D1	D0
8-bit raw or DPCM 14-8-14/12-8-12/10-8-10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D7	D6	D5	D4	D3	D2	D1	D0
10-bit raw or DPCM 14-10-14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

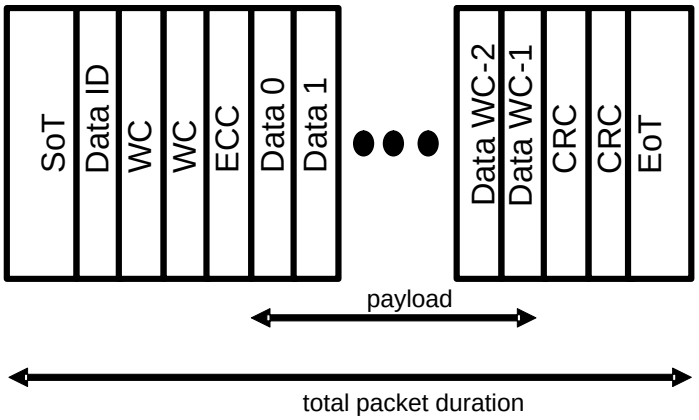
Format Name		23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
12-bit raw		0	0	0	0	0	0	0	0	0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
14-bit raw		0	0	0	0	0	0	0	0	0	0	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
8-bit arbitrary/embedded		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D7	D6	D5	D4	D3	D2	D1	D0
16-bit RGB (RGB565)		B4	B3	B2	B1	B0	0	0	0	G5	G4	G3	G2	G1	G0	0	0	R4	R3	R2	R1	R0	0	0	0
15-bit RGB (RGB555)		B4	B3	B2	B1	B0	0	0	0	G4	G3	G2	G1	G0	0	0	0	R4	R3	R2	R1	R0	0	0	0
24-bit RGB (RGB888)		B7	B6	B5	B4	B3	B2	B1	B0	G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0
12-bit RGB (RGB444)		B3	B2	B1	B0	0	0	0	0	G3	G2	G1	G0	0	0	0	0	R3	R2	R1	R0	0	0	0	0
18-bit RGB (RGB666)		B5	B4	B3	B2	B1	B0	0	0	G5	G4	G3	G2	G1	G0	0	0	R5	R4	R3	R2	R1	R0	0	0
YUV422 8-bit	1st CLK	V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0	Y07	Y06	Y05	Y04	Y03	Y02	Y01	Y00
	2nd CLK	V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
YUV422 10-bit	1st CLK	V9	V8	V7	V6	V5	V4	V3	V2	U9	U8	U7	U6	U5	U4	U3	U2	Y09	Y08	Y07	Y06	Y05	Y04	Y03	Y02
	2nd CLK	V9	V8	V7	V6	V5	V4	V3	V2	U9	U8	U7	U6	U5	U4	U3	U2	Y19	Y18	Y17	Y16	Y15	Y14	Y13	Y12
YUV420 8-bit (legacy)	Odd Line[2] 1st CLK	0	0	0	0	0	0	0	0	U7	U6	U5	U4	U3	U2	U1	U0	Y07	Y06	Y05	Y04	Y03	Y02	Y01	Y00
	Odd Line 2nd CLK	0	0	0	0	0	0	0	0	U7	U6	U5	U4	U3	U2	U1	U0	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
	Even Line 1st CLK	V7	V6	V5	V4	V3	V2	V1	V0	0	0	0	0	0	0	0	0	Y07	Y06	Y05	Y04	Y03	Y02	Y01	Y00
	Even Line 2nd CLK	V7	V6	V5	V4	V3	V2	V1	V0	0	0	0	0	0	0	0	0	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
YUV420 8-bit (CSPS)	Odd Line	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
	Even Line 1st CLK	V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0	Y07	Y06	Y05	Y04	Y03	Y02	Y01	Y00
	Even Line 2nd CLK	V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
YUV420 10-bit (CSPS)	Odd Line	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Y9	Y8	Y7	Y6	Y5	Y4	Y3	Y2
	Even Line 1st CLK	V9	V8	V7	V6	V5	V4	V3	V2	U9	U8	U7	U6	U5	U4	U3	U2	Y9	Y8	Y7	Y6	Y5	Y4	Y3	Y2
	Even Line 2nd CLK	V9	V8	V7	V6	V5	V4	V3	V2	U9	U8	U7	U6	U5	U4	U3	U2	Y9	Y8	Y7	Y6	Y5	Y4	Y3	Y2

1. The bits marked as '0' are actually don't cares.
2. Odd Line refers to the first line from the CSI unit.

26.5 CSI Packet Structure

The CSI packet structure is illustrated in Figure 88. It has a start-of-transmission sequence (SoT), which contains a 1-byte preamble sequence. The packet is composed of a 4-byte header, a body of word count (WC) bytes, and a 2-byte CRC check. The end-of-transmission (EoT) is indicated by the line state going from high-speed transmission to the LP11 state.

Figure 88: CSI Packet Structure



26.6 CSI Implementation

The CSI interface consists of:

- MIPI D-PHY block
- Control Interface Logic (CIL) for Serial Clock Receiver
- CSI datapath module which is implemented as a sub-module of VI module

In the Tegra K1 implementation, three CSI bricks including the csiab_pad with 4x data lanes, the shared dsib_pad with 4x data lanes, and the csie_pad with 1x data lanes receive serial data from cameras, deserialize them into 8-bit parallel data, and send them to the CSICIL block. The CSICIL block performs packet boundary detection by searching for the packet preamble. The main CSI module receives an 8-bit packet aligned data from CSICIL, performs parity checking on the header, header decoding, CRC check, and pixel parsing. The output is sent out to the VI through the 28-bit bus, which outputs 1 pixel per VI clock.

26.7 Performance Limitations

The performance of the CSI interface is subject to several factors that must all be met. The performance factors that are directly related to the CSI interface are the serial data rate and the internal pixel rate. Maximum serial data rate is expected to be 1Gbps given the ideal system condition. This data rate may be degraded depending on system design and may also be limited by the CSI transmitter in the camera.

For each CSI data lane, the serial data stream received by the CSI PHY is internally converted to a byte stream by the CSI Control Interface Logic (CIL) before further processed by the CSI module that resides as part of the Video Input (VI) module. The CSI byte clock is used to transfer this data byte stream (1 byte per data lane) between the CIL and the CSI datapaths. This CSI byte clock is derived from the received CSI serial clock by dividing the CSI serial clock by 4. So for 1 Gbps serial CSI data rate, the serial clock frequency is 500 MHz and the byte clock frequency is 125 MHz.

The byte stream received by the CSI CIL logic is converted to a pixel stream at 1 pixel per clock in the CSI module. This pixel stream output is further processed by other modules (ISP or the rest of the VI datapath). Depending on where the CSI pixel stream is processed (ISP or VI), the maximum pixel clock frequency is limited by the pixel clock frequency of the modules that process this pixel stream.

Note:	VI/ISP pixel frequency may be lower than CSI output pixel clock frequency and therefore, may limit the actual pixel data rate. Please refer to the ISP and VI specifications.

Note that with multiple data lanes per CSI interface, the maximum serial data link speed may not be achievable due to pixel clock frequency limitations. Also, if two data streams come from the same CSI interface, since the streams are interleaved in time, the pixel clock frequency limitations may also limit the frame rate of the combined streams.

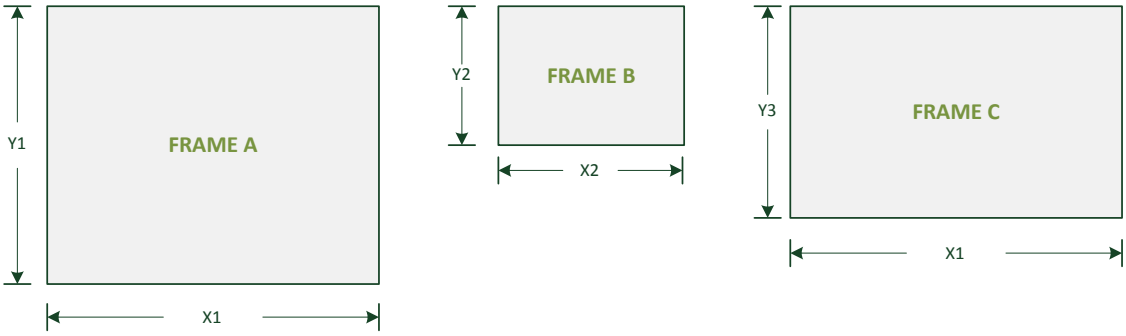
Table 107: Packet Efficiency

	Best* WC=65536	Typical* WC=1024
Payload	524 µs	8192 ns
Overhead	356 ns	356 ns
Efficiency	99.9%	96%

26.8 Frame Size Mismatch Scenarios

Several frame mismatch scenarios can occur during image capture. In these situations, a frame might not have the expected dimensions as shown in the following figure.

Figure 89: Frame Size Variations



Frame A = Full Resolution Image of size (X1,Y1)
Frame B - Preview image of size (X2,Y2) where, Y2 < Y1 & X2 < X1
Frame C - Preview image of size (X1,Y3) where, Y3 < Y1

The following subsections describe the behavior of the CSI (and VI/ISP) in the various frame size mismatch scenarios:

26.8.1 Received Frame Width and Height Matches Programmed Width and Height

When the CSI receives any frame size, it starts capturing the frame. The VI will raise the sync point when it receives the EoF at the right time.

26.8.2 Received Frame Width Does Not Match Programmed Width

Packet Width Mismatch Error

If the CSI receives any frame size in which the width of the incoming frame does not match the programmed value, the error is detected and signaled to the VI. The VI will drop the frame in the case of packet width mismatch errors.

26.8.3 Received Frame Height Does Not Match Programmed Height

CASE 1: Frame Height Mismatch Error - Fewer Rows Than Expected

Example: The CSI is instructed to capture Frame A:

When the sensor receives Frame C, it starts capturing the frame as the width of the frame matches the expected frame width. The CSI will encounter an early EoF. At this time, it will rearm the pixel parser state-machine with the same set of frame capture parameters and wait for the next frame.

If this frame was to be sent to memory, the VI unit will not detect that it is not the right frame and will not raise the sync point. When it receives the next frame from the CSI, it will overwrite the memory buffer (at the buffer location pointed by previous frame parameter).

If this frame was to be sent to the ISP, the VI forwards the EoF control packet to the ISP and indicates that the Frame was too short. The CSI and VI will not pad the frame to make the frame the right size. The ISP unit needs to detect the error condition and should not update the parameter for the next frame and apply the same parameters (as the previous frame) for the next incoming frame from the VI.

CASE 2: Frame Height Mismatch Error - More Rows Than Expected

Example 2: The CSI is instructed to capture Frame C:

When the CSI receives Frame A, it starts capturing the frame as the width of the frame is matched.

For the frame to be sent to memory, the VI marks the last atom of the frame to the memory as a non-posted write. The VI then raises a sync point when the WrAck and EoF at the expected frame size if received. In this case, instead of EoF, the VI will receive more pixel data. On detecting this condition, the VI will not raise the sync point; it drops the rest of the frame and does not corrupt the memory buffer. The CSI will rearm the pixel parser state machine with the same set of frame capture parameters and waits for the next frame. Because the VI is also aware of the expected frame height and detects the frame to be erroneous, it will overwrite the memory buffer with the next frame.

In case this frame was to be sent to the ISP, the VI, when it receives the regular pixel data instead of EoF flag from the CSI, will drop the data and generate an EoF flag to the ISP to terminate the frame and mark it as frame-too-big. The ISP unit needs to detect the error condition; it should not update the parameter for the next frame and will apply the same parameters (as the previous frame) for the next incoming frame from the VI.

26.8.4 Double Buffering of the Configuration Registers

There are two copies of configuration registers: operational and the shadow registers. The shadow registers are updated from the Host1x input when it does not have a pending capture command. The shadow register is copied to the operational register when a frame is successfully captured (at the EoF received at the right time). In the event of a frame size mismatch, the shadow registers are not copied and the current settings remain active for the next incoming frame:

1. **Successful Frame Capture:** The CSI pixel parser (and header parser) sub-units copy the shadow configuration register to the operational configuration register, if it needs to capture the next frame. This configurations need to move along the pipeline with the SoF marker.
2. **Failed Frame Capture (Dropped):** When a frame is dropped for being too big or too small, the shadow registers are not copied over to the operational register and the CSI remains armed for the next frame.

Whenever the CSI drops an incoming frame, whether due to a mismatch in virtual channel, data type, width, or the height, this information is appropriately logged in the status register. This would help in debugging any system issues where the software arms the CSI to capture the frame but eventually times out due to a missing sync point. There is only one copy of this register (per CSI channel) and every subsequent dropped frame will update this status register.

26.9 Error Handling

The CSI is required to tolerate and/or recover stream errors at various levels.

Error Categories	Classification	Type	Description
D-PHY Level Errors	Start of Transmission (SoT) Error	Single-bit error	Corrected an error condition signal from PHY/CIL
		Multi-bit error	Packet discarded
	Control Error	Incorrect line state sequence	XError signal asserted
	Escape Entry Error	Escape command different from 8-bits and escape handshaking incorrect	XError signal asserted
	End of Transmission (EoT) Error	A bit of the payload not on a byte boundary	Unrecoverable error condition signaled to the application layer
Packet Level Errors	Packet Header Error	Single bit error	Detected and corrected by the ECC code
		Multi-bit error	Detected but not corrected; error is flagged and interrupt generated
	Packet Payload Errors	Signaled through CRC Code	Detected but not corrected; error is flagged and interrupt generated
Protocol Decoding Errors	Frame Sync Error	FE is not matched to FS	Frame data is supplied to the VI and an error is signaled as a stream error to the application layer.
	Unrecognized ID	Data ID Error	Packet discarded
CSI Block Specific Errors	Packet Header Mismatch Error	Data ID Error	If the virtual channel ID does not match the programmed value for the corresponding pixel parser then the packet is discarded
		Data Type Error	If the data-type does not match the programmed value for the corresponding pixel parser then the packet is discarded
	FIFO Overflow	Backpressure	Frame is dropped
	Packet Word Count Mismatch	Packet longer than word count	Packet is dropped
		Packet shorter than word count	Packet is dropped
	Frame Height Mismatch	Frame larger than programmed size	Frame data is supplied to the VI until EoF and then the pixel parser is re-armed. VI will receive or drop the additional data depending on destination (MEMC or ISP)
		Frame smaller than programmed size	The pixel parser is rearmed on EoF and waits for the next frame
	Padding Congestion	Data received during Frame padding	Padding removed for Tegra K1

26.10 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, this embedded data may be accidentally processed in VI/ISP. Currently, VI/ISP cannot differentiate between embedded data or pixel data. Refer to the VI chapter for more information on the options for managing embedded data.

26.11 CSI Datapath Module

The CSI datapath consists of two input ports and two output ports. There are two datapaths for pixel stream processing; therefore, at any time, a maximum of two input ports may be active simultaneously.

The components of the CSI datapath include:

- Two input ports from: CSI A interface and CSI B interface. Each port is capable of carrying a stream with CSI compatible data format.
- Two asynchronous input buffers (FIFOs) to receive streams from the CSI A interface and the CSI B interface.
- Three header parsers for searching packet header and to perform error detection and correction of CSI header.
- Two pixel parsers with corresponding output ports. Each pixel parser can convert a CSI packet data stream to output a pixel stream. Each output port consists of a maximum 24 bits of data per clock. Depending on the output data format options, one or two pixels per clock may be sent.

26.11.1 Header Parser

CSI pixel stream processing mainly consists of header parser and pixel parser. There are 3 header parsers: header parser A, header parser B, and header parser H.

Header parser A is enabled when CSI A interface is selected as input source. Similarly, header parser B is enabled when CSI B interface is selected as input source.

In general the header parser is used for:

- Detecting long and short packet headers and deciding what to do with the packet. This includes performing error detection and correction on the packet header prior to making decision and dealing with uncorrectable packet header.
- Skipping extra data on longer than expected data packets.
- Skipping next packet on imminent input buffer overflow when the pixel parser is busy processing current packet.

When enabled, the header parser will interpret CSI packet header, perform error detection and correction. If the packet header is uncorrectable, then an error is flagged and interrupt generated. If good or correctable CSI packet header is found, then the header parser will check the Data Identifier (DI) byte in the packet header and check the 2-bit Virtual Channel (VC) identifier and the 6-bit Data Type (DT) within this Data Identifier byte and will also check the 2-byte Word Count (WC) value in the packet header. If the virtual channel ID does not match the expected (programmed) value for the corresponding pixel parser then the packet is discarded. If the virtual channel ID matches the expected (programmed) value for the corresponding pixel parser then the header parser must decide what to do with the packet depending on the Data Type and the Word Count value. For long packet, if the Data Type does not match the expected (programmed) value for the corresponding pixel parser then the packet is discarded. If the Data Type of the long packet matches the expected (programmed) value for the corresponding pixel parser, then the corresponding pixel parser is notified to process the remaining packet data and checksum.

When first enabled, the header parser must always search for Frame Start packet which is a CSI short packet. When frame start packet/signal is found, the header parser will notify the corresponding pixel parser so that it can prepare to process a new frame and output frame start code.

26.11.1.1 Data Type

There are 64 possible data types based on the 6-bit Data Type value.

Data Type	Description
0x00 – 0x07	Synchronization Short Packet Data Types
0x08 – 0x0F	Generic Short Packet Data Types
0x10 – 0x17	Generic Long Packet Data Types
0x18 – 0x1F	YUV Data
0x20 – 0x27	RGB Data
0x28 – 0x2F	RAW Data
0x30 – 0x37	User Defined Byte-based Data
0x38 – 0x3F	Reserved

26.11.1.2 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

26.11.1.3 Long Packets

CSI long packets are data packets which always consist of 1 line of data per packet with exception of arbitrary data packets. There are various data types defined. Please refer to the Input Data Format section for a summary of CSI data formats and their corresponding CSI module internal output format.

Generic long packets need special handling. There are 3 generic long packet types defined: null packet (DT=0x10), blanking data packet (DT=0x11), and embedded data packet (DT=0x12). There are also 5 reserved generic long packet types (DT=0x13 to 0x17). The header parser should discard all null packets and all reserved generic long packets.

Blanking data packets may contain blank color or blank image information. The transmission of blanking data packet is optional in the CSI2.0 specification. Null and blanking data are defined in the CSI2.0 specification mainly to accommodate a receiver which is timing sensitive and requires line start/end for every line in the frame. As a CSI receiver, this product does not have such blank timing sensitivity and it typically does not process blank data and does not store it to memory. Blanking data packet can therefore be discarded by the header parser.

However, some module may have a requirement for specific number of “blank” or extra lines at the end of vertical active area, such as the ISP module, which requires input active scan area to be about 6 lines larger than output active area. If a module that takes CSI output stream requires extra lines at the end of active image then blanking packet data can be used to generate these additional lines. In the future, there might be other reasons to process blank data. So, an option should be provided to accept and process blanking data. In the absence of better definition of blanking data format in the CSI specification, if software decides that blanking data cannot be discarded then, the pixel parser should assume that the blanking data format is the same as the programmed expected data type of the pixel stream.

The current CSI2.0 specification specifies that embedded data packets consists of 8-bit arbitrary data. This embedded data is, typically not the same pixel color as the image data. It might be used to send headers/status for JPEG/MPEG compressed data at the beginning or end of image data or in between image data lines. It might also be used to send closed caption text information or other type of information.

The exact use of embedded data is not clear, and therefore in most cases, embedded data is probably not used, or if sent by the transmitter, the embedded data packets can simply be discarded. If for some reason, it is important to receive the embedded data, there are other options that should be supported. Since there are two pixel parsers, if there is only one video source, it is best to direct embedded data to the other pixel parser which does not process the normal image data. In this way, the embedded data can be treated the same way as 8-bit arbitrary/compressed image data and can be output in 8-bit/clock or 16-bit/clock format. If there are two video sources that occupy both the pixel parsers, then embedded data if it has to be stored to memory, needs to be output in the same channel as the image data as 8-bit/16-bit data. But this also means that the module that receives output of CSI module and does image processing must have the ability to differentiate embedded data and not do image processing on this data prior to writing it to memory.

26.11.1.4 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.

26.12 Test Pattern Generator (TPG)

The test pattern generator is a configurable resource introduced to improve hardware verification capability for the Tegra CSI. Because a provision for inserting pixel data from the Host interface is not available, the hardware verification of the CSI and VI unit is difficult. The only sources of pixel data that can exercise the CSI are image sensors. In general, the sensor data cannot be controlled, making any self-checking test impossible. Test pattern generation modes offered by sensors are also ineffective because they have limited control.

There are two separate test pattern generators that can be configured to provide for the generation of synthetic image data, which is delivered to the PPA and PPB input FIFOs. The image data is multiplexed into the CSI data patch between lane-merging logic and the data FIFOs. Programmable configuration parameters are available to allow variation in pixel data, data type, and spacing between various packets. In the real system, the LP state transition between any two CSI packets guarantees some idle time between the lines. This spacing between the lines from the pattern generator should be programmed to a minimum of 8 cycles, as the later stages of VI expects some idle cycles between the lines in order for it not to overflow.

A difference engine architecture is employed which can generate almost arbitrary patterns that consist of frequency sweeps or intensity ramps in both the horizontal and vertical directions. In addition, by using selected bits from the difference engine variables as addresses, a color patchwork pattern can be generated. The output from the difference engine is modeled off of a second-order partial differential equation of the form:

$$\varphi_{out} = \varphi + x \cdot \frac{d\varphi}{dx} + x^2 \cdot \frac{d^2\varphi}{dx^2}$$

Where:

$$\varphi = PHASE$$

$$\frac{d\varphi}{dx} = FREQ$$

$$\frac{d^2\varphi}{dx^2} = FREQ\_RATE$$

The PHASE, FREQ, and FREQ_RATE variables may be programmed to create a variety of different test patterns and are either used directly or as an index into a look-up table (LUT). Table 108 summarizes the configuration parameters for the TPG.

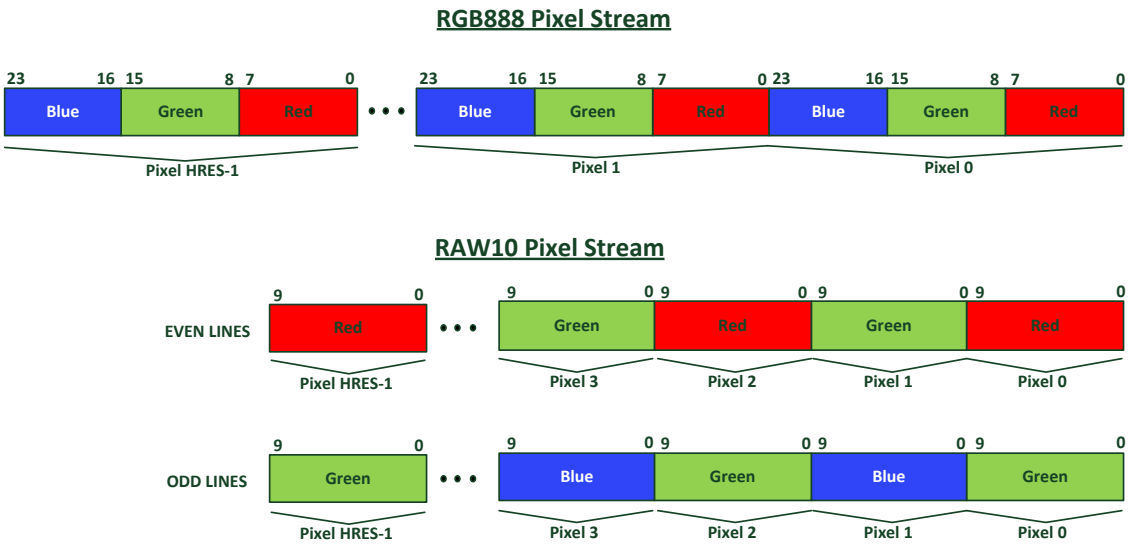
Table 108: Description of TPG Configuration Parameters

Configuration Variables for CSI_0 and CSI_1 Test Pattern Generators	TPG Frame Configuration	Image Height	Lines per frame
		Image Width	Pixels per line
		Vertical Blanking	Blank lines between frames
		Horizontal Blanking	Blank pixels between lines
		Initial Phase	Initial phase
		Mode	Direct or color patch
		Format	RGB888 or RAW10 support
	TPG Channel Configurations (R, G, and B)	Vertical Initial Frequency	Initial frequency
		Horizontal Initial Frequency	Initial frequency
		Vertical Frequency Rate	Rate of change for vertical frequency
		Horizontal Frequency Rate	Rate of change for horizontal frequency

26.12.1 TPG Frame Configuration Parameters

The TPG can be configured to provide the test images in either an RGB888 or Bayer RAW10 data format. The RGB888 data is delivered to the lane merging logic as a sequence of bytes representing the RGB triads for each pixel in an HRESxVRES image frame. For the Bayer RAW10 data, the color components of the RGB pixel are sampled in an alternating fashion, depending on whether the current row and pixel within the row is even or odd.

Figure 90: RGB888 and RAW10 TPG Pixel Streams



Note: Byte ordering for RGB888 and RAW10 on the CSI2VI interface follows the same convention as data received over the MIPI interface.

The data from the test pattern generator has the same format (with the packet header, CRC, ECC, etc.) as received on the output of lane-merging logic during normal operation.

26.12.2 Modes of Operation

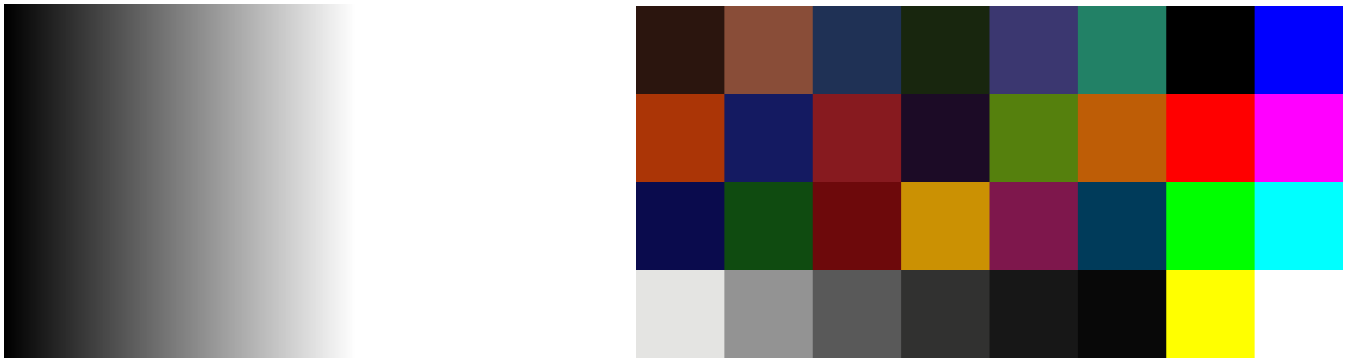
The test pattern generator operates in two modes:

- Direct The output of the difference engine computation will be viewed directly.
- Color Patch The difference engine output will be used to generate a patchwork of colors.

In Direct mode, the output comes directly from the value of the phase accumulator in the difference engine. An example of the kind of pattern that can be made is shown in Figure 91(a). In Color Patch mode, the output comes from indexing a 32-element LUT.

The various bits from both the horizontal and vertical phase accumulators are used as address bits into a LUT of standard color values, allowing a patchwork of color test patterns to be generated. The colors in the pattern consist of linear RGB space representations of the 24 “Macbeth” test chart colors, shown on the left side of Figure 91(b), and the 8 “100% saturation” colors of a TV test pattern, on the right side of Figure 91(a). This gives a total of 32 color patches.

Figure 91: Example of (a) Direct Mode (b) Color Patch Generator Output



(a) (b)

26.13 Differential Pulse Code Modulation Support

The use of Differential Pulse Code Modulation (DPCM) compression schemes to reduce the data bandwidth requirements have been approached in both the SMIA and MIPI camera serial interface standards. In the MIPI CSI2 rev 1.1 a new data type for compressed data was introduced. In this format, RAW-10 and RAW-12 data can be transmitted as 6/7/8 bits per pixel. The SMIA also introduces a compression for RAW14 data providing transmission as 8/10 bits per pixel. The DPCM support is an optional feature for the CSI-2 and the data compression is not mandated. However, if data compression is used, it must be implemented as described in annex E of the MIPI CSI-2 specification v01-01-00.

The MIPI and SMIA data compression schemes use an X–Y–Z naming convention, where:

- X is the number of bits per pixel in the original image
- Y is the encoded (compressed) bits per pixel
- Z is the decoded (uncompressed) bits per pixel.

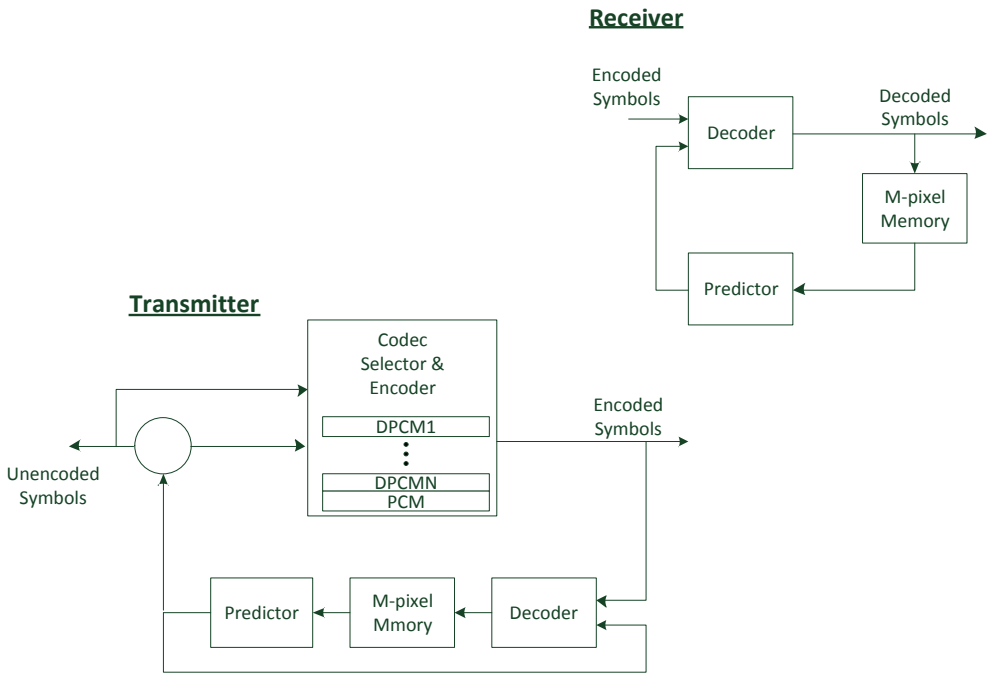
The following data compression schemes are defined for the Tegra K1 CSI unit:

- 14-bit RAW Compression Support (SMIA): 14-8-14; 14-10-14
- 12-bit RAW Compression Support (MIPI): 12–8–12; 12-7-12; 12-6-12
- 10-bit RAW Compression Support (MIPI): 10–8–10; 10-7-10; 10-6-10

To identify the type of data on the CSI-2 interface, packets with compressed data have a user-defined data type value. The encoder uses a simple algorithm to encode the pixel values. A fixed number of pixel values at the beginning of each line are

encoded without using prediction. These first few values are used to initialize the predictor block. The remaining pixel values on the line are encoded using prediction. The following figure illustrates the data compression system for the transmitter and receiver.

Figure 92: Data Compression System Block Diagram



26.14 Software Requirements

26.14.1 Error Counters

To help debug, three 32-bit error counters have been implemented. Software can program each counter to increment at the event of any one of many error conditions or status change. The error conditions are:

- Header errors corrected
- Header uncorrectable errors
- CRC errors
- Packets too short, actual length less than WC
- Overflow errors due to padding packets too short
- FIFO overflow errors
- Illegal word count
- SoT single-bit error
- SoT multi-bit error, packet discarded
- EoT sequence error
- LP-CTRL error
- The statistics that can be monitored include:
- Line packets processed
- Total packets processed

26.14.2 Programming Sequence

The following sequence is recommended for capturing a single frame:

1. Set up CSI registers for use case such as number of lanes, virtual channel, etc.
2. Initialize and power up CSI interface
3. Wait for initialization time or done signal from calibration logic
4. Power up camera through the I^2C interface
5. All CSI data and clock lanes are in stop state, LP11
6. Initiate frame capture through the I^2C
7. Frame done, CSI goes back to stop state, LP11

26.14.3 Escape Mode Handling

In the MIPI PHY specification, an escape mode is available for low speed command and data transfer. In our implementation, the escape mode entry is detected inside CSICIL. The command byte will be deposited in a register and interrupt generated from CSI. Software reads the register and decodes. CSI supports only one escape mode command which is Ultra-Low Power Mode (ULPM).

ULPM Command

Upon receiving the ULPM, software will synchronously power down the CSI interface and the camera as follows:

1. Software puts the camera in sleep mode through I^2C
2. The camera sends a ULPM command to CSI
3. CSI receives the ULPM command and raises an interrupt
4. Software powers down CSI, which goes to sleep in sync with the camera's CSI transmitter

26.15 DPHY Modes of Operation

The MIPI DPHY has 3 basic modes of operation. High speed mode employs differential signaling and achieves data rate of 1 Gbps. Both the driver and receiver are matched to 100 ohms differential. The driver is voltage mode for lower power consumption as opposed to current mode.

Low power control mode employs single-ended CMOS signaling for handshaking between camera and host. In this mode, there is no clock and no maximum symbol time defined in the specification. The receiver samples the input using both edges of the internal pixel clock, so the timing resolution is half the clock period of the pixel clock period.

Table 109: MIPI DPHY Mode of Operations

Modes	Description	Clock
High speed (HS)	High speed differential signaling. Up to 1.5 Gbps. Burst transmission for low power.	750 MHz differential
Low Power (LP) Control	Single-ended 1.2V CMOS level. Low speed signaling for handshaking. Supports ULPM sleep state.	No Clock

26.16 MIPI-CSI Registers

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

26.16.1 CSI_INPUT_STREAM_A_CONTROL_0

CSI Input Stream A Control

Offset: 0x20e | Byte Offset: 0x838 | Read/Write: R/W | Reset: 0x00ff0004 (0bxxxxxxx011111111xxxxxxxxxx0x100)

Bit	Reset	Description
24:16	0xff	CSI_A_SKIP_PACKET_THRESHOLD: CSI-A Skip Packet Threshold. This value is compared against the internal FIFO that buffer the input streams. A packet will be skipped (discarded) if the pixel stream processor is busy (probably due to padding process of a short line) and the number of entries in the internal FIFO exceeds this threshold value. Note that each entry in the internal FIFO buffer is four bytes. To turn off this feature, set the value to its maximum value (all ones).
4	0x0	CSI_A_SKIP_PACKET_THRESHOLD_ENABLE: Enables skip packet threshold feature. Skip packet feature is enabled. 0 = DISABLE: Skip packet feature is disabled. 1 = ENABLE
2	0x1	CSI_A_BYPASS_ALIGN: Bypass aligning CSIA and CSIB lanes if the valids for the lanes are out of sync by one cycle
1:0	0x0	CSI_A_DATA_LANE: CSI-A Data Lane. 0= 1 data lane; 1= 2 data lanes; 2= 3 data lanes; 3= 4 data lanes Note: Three data lanes are not supported in Test Pattern Generation mode.

26.16.2 CSI_PIXEL_STREAM_A_CONTROL0_0

CSI Pixel Stream A Control 0

Offset: 0x20f | Byte Offset: 0x83c | Read/Write: R/W | Reset: 0xXXXX01X0 (0bxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx000)

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. 0 = PAD0S: Lines of all zeros will be used to pad out frames that are shorter than expected height. PPA_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this field is set to PAD0S. 1 = PAD1S: Lines of all ones will be used to pad out frames that are shorter than expected height. PPA_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this fields is set to PAD1S. 2 = NOPAD: Short frames will not be padded out.
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. 0 = ENABLE: Single bit errors in the header will be automatically corrected. 1 = DISABLE: 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).

Bit	Reset	Description
25:24	X	CSI_PPA_PAD_SHORT_LINE: CSI Pixel Parser A Pad Short Line. This specifies how to deal with shorter than expected line (the number of bytes received is less than the specified word count). 0 = PAD0S: short line is padded by pixel of zeros such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet may be discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 1 = PAD1S: short line is padded by pixel of ones such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet may be discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 2 = NOPAD: A short line is not padded (will output less pixels than expected). This option is not recommended and may cause other modules that receives CSI output stream to hang up.
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. 0 = DISCARD: discard (throw away) embedded data 1 = EMBEDDED: Output embedded data as an 8-bpp arbitrary data stream.
19:16	X	CSI_PPA_OUTPUT_FORMAT_OPTIONS: CSI Pixel Parser A Output Format Options. This parameter specifies options for output data format. 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: 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.
8	0x1	CSI_PPA_WC_CHECK: CSI Pixel Parser A Data WC Check. This parameter specifies whether the word count is checked. 0 = DISABLE : Wordcount Check is disabled 1 = ENABLE: Wordcount Check is enabled.
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. 0 = DISABLE: Data CRC Check is disabled regardless of whether there are CRC checksum at the end of the packet. 1 = ENABLE: Data CRC Check is enabled.
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. 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. Interlaced support relies on frame number and will not work in REGISTER mode. 1 = HEADER: 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. It is recommended to always program this bit to 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. 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: Enabled - Data Identifier byte in packet header should be compared against the CSI_PPA_DATA_TYPE and the CSI_PPA_VIRTUAL_CHANNEL_ID.
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. 0 = NOT_SENT: Packet header is not sent. This setting should not be used if the stream

Bit	Reset	Description
		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: Packet header is sent. This setting should be used if the stream source is CSI Interface A or B.
2:0	0x0	CSI_PPA_STREAM_SOURCE: CSI Pixel Parser A Stream Source 0 = CSI_A: CSI Interface A 1 = CSI_B: CSI Interface B

26.16.3 CSI_PIXEL_STREAM_A_CONTROL1_0

CSI Pixel Stream A Control 1

Offset: 0x210 | Byte Offset: 0x840 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)|

Bit	Reset	Description
7:4	0x0	CSI_PPA_TOP_FIELD_FRAME_MASK: CSI Pixel Parser A Top Field Frame Mask
3:0	0x0	CSI_PPA_TOP_FIELD_FRAME: CSI Pixel Parser A Top Field Frame. This parameter specifies the frame number for top field detection for interlaced input video stream. Top Field is indicated when each of the least significant four bits of the frame number that has a one in its mask bit matches the corresponding bit in this parameter. In other words, Top Field is detected when the bitwise OR 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. Due to the reliance on WC field in packet header, interlaced support will not work if WC is selected from REGISTER mode.

26.16.4 CSI_PIXEL_STREAM_A_GAP_0

CSI Pixel Stream A Gap

Offset: 0x211 | Byte Offset: 0x844 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PPA_FRAME_MIN_GAP: Minimum number of viclk cycles from end of frame (Video_control = EF) to start of next frame (Video_control = SF). This parameter ensures that the minimum V-blank time requirement of VI/ISP is satisfied. This field takes effect only when the frame gap of the input stream is less than the specified value.
15:0	0x0	PPA_LINE_MIN_GAP: Minimum number of viclk cycles from end of previous line (Video_control = EL_DATA) to start of next line (Video_control = SL). This parameter ensures that the minimum H-blank time requirement of VI/ISP is satisfied. This field takes effect only when the line gap of the input stream is less than the specified value.

26.16.5 CSI_PIXEL_STREAM_PPA_COMMAND_0

CSI Pixel Parser A Command

Offset: 0x212 | Byte Offset: 0x848 | Read/Write: R/W | Reset: 0x0000XXX4 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx100)

Bit	Reset	Description
15:12	X	CSI_PPA_START_MARKER_FRAME_MAX: CSI Pixel Parser A Start Marker Maximum. Start Frame is indicated when the Minimum condition above is met and the least significant four bits of the frame number are less than or equal to this value.
11:8	X	CSI_PPA_START_MARKER_FRAME_MIN: CSI Pixel Parser A Start Marker Minimum. Start Frame is indicated when the Maximum condition below is met and the least significant four bits of the frame number are greater than or equal to this value.

Bit	Reset	Description
4	X	CSI_PPA_VSYNC_START_MARKER: CSI Pixel Parser A VSYNC Start Marker. The start of frame is indicated when the VSYNC signal is received. When the input stream is from the VIP path and the data mode is PACKET, then this field may be programmed to VSYNC. 0 = FSPKT: Start of frame is indicated when a Frame Start short packet is received with a frame number whose least significant four bits are greater than or equal to CSI_PPA_START_MARKER_FRAME_MIN and less than or equal to CSI_PPA_START_MARKER_FRAME_MAX. When the input stream is from a CSI port, or from the host path, or from the VIP path and the data mode is PAYLOAD_ONLY, then this field may be programmed to FSPKT. 1 = VSYNC
2	0x1	CSI_PPA_SINGLE_SHOT: CSI Pixel Parser A Single Shot Mode. Software should clear it along with disabling the CSI_PPA_ENABLE, once a frame is captured 0 = DISABLE 1 = ENABLE
1:0	0x0	CSI_PPA_ENABLE: CSI Pixel Parser A Enable. This parameter controls CSI Pixel Parser A to start or stop receiving data. Reset (disable immediately) Enabling the pixel parser does not enable the corresponding input source to receive data. If the pixel parser is enabled later than the corresponding input source, the CSI will keep on rejecting incoming stream, until it encounters a valid SF. 0 = NOP: no operation 1 = ENABLE: enable at the next frame start as specified by the CSI Start Marker 2 = DISABLE: disable after current frame end and before next frame start. 3 = RST

26.16.6 CSI_PIXEL_STREAM_A_EXPECTED_FRAME_0

CSI Pixel Stream A Expected Frame

Offset: 0x213 | Byte Offset: 0x84c | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
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.

26.16.7 CSI_CSI_PIXEL_PARSER_A_INTERRUPT_MASK_0

CSI Pixel Parser A Interrupt Mask

Offset: 0x214 | Byte Offset: 0x850 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx0xx000000000000)

Bit	Reset	Description
14	0x0	HPA_UNC_HDR_ERR_INT_MASK: Interrupt Mask for HPA_UNC_HDR_ERR. 0 = DISABLED : Do not generate an interrupt when HPA_UNC_HDR_ERR is set. 1 = ENABLED: Generate an interrupt when HPA_UNC_HDR_ERR is set.
10	0x0	PPA_SPARE_STATUS_1_INT_MASK: Interrupt Mask for PPA_SPARE_STATUS_1. 0 = DISABLED : Do not generate an interrupt when PPA_SPARE_STATUS_1 is set. 1 = ENABLED: Generate an interrupt when PPA_SPARE_STATUS_1 is set.
9	0x0	PPA_INTERFRAME_LINE_INT_MASK: Interrupt Mask for PPA_INTERFRAME_LINE. 0 = DISABLED : Do not generate an interrupt when PPA_INTERFRAME_LINE is set. 1 = ENABLED: Generate an interrupt when PPA_INTERFRAME_LINE is set.
8	0x0	PPA_EXTRA_SF_INT_MASK: Interrupt Mask for PPA_EXTRA_SF. 0 = DISABLED : Do not generate an interrupt when PPA_EXTRA_SF is set.

Bit	Reset	Description
		1 = ENABLED: Generate an interrupt when PPA_EXTRA_SF is set.
7	0x0	PPA_SHORT_FRAME_INT_MASK: Interrupt Mask for PPA_SHORT_FRAME. 0 = DISABLED : Do not generate an interrupt when PPA_SHORT_FRAME is set. 1 = ENABLED: Generate an interrupt when PPA_SHORT_FRAME is set.
6	0x0	PPA_STMERR_INT_MASK: Interrupt Mask for PPA_STMERR. 0 = DISABLED : Do not generate an interrupt when PPA_STMERR is set. 1 = ENABLED: Generate an interrupt when PPA_STMERR is set.
5	0x0	PPA_FIFO_OVRF_INT_MASK: Interrupt Mask for PPA_FIFO_OVRF. 0 = DISABLED : Do not generate an interrupt when PPA_FIFO_OVRF is set. 1 = ENABLED: Generate an interrupt when PPA_FIFO_OVRF is set.
4	0x0	PPA_PL_CRC_ERR_INT_MASK: Interrupt Mask for PPA_PL_CRC_ERR. 0 = DISABLED : Do not generate an interrupt when PPA_PL_CRC_ERR is set. 1 = ENABLED: Generate an interrupt when PPA_PL_CRC_ERR is set.
3	0x0	PPA_SL_PKT_DROPPED_INT_MASK: Interrupt Mask for PPA_SL_PKT_DROPPED. 0 = DISABLED : Do not generate an interrupt when PPA_SL_PKT_DROPPED is set. 1 = ENABLED: Generate an interrupt when PPA_SL_PKT_DROPPED is set.
2	0x0	PPA_SL_PROCESSED_INT_MASK: Interrupt Mask for PPA_SL_PROCESSED. 0 = DISABLED : Do not generate an interrupt when PPA_SL_PROCESSED is set. 1 = ENABLED: Generate an interrupt when PPA_SL_PROCESSED is set.
1	0x0	PPA_ILL_WD_CNT_INT_MASK: Interrupt Mask for PPA_ILL_WD_CNT. 0 = DISABLED : Do not generate an interrupt when PPA_ILL_WD_CNT is set. 1 = ENABLED: Generate an interrupt when PPA_ILL_WD_CNT is set.
0	0x0	PPA_HDR_ERR_COR_INT_MASK: Interrupt Mask for PPA_HDR_ERR_COR. 0 = DISABLED : Do not generate an interrupt when PPA_HDR_ERR_COR is set. 1 = ENABLED: Generate an interrupt when PPA_HDR_ERR_COR is set.

26.16.8 CSI_CSI_PIXEL_PARSER_A_STATUS_0

Pixel Parser A Status

These status bits are cleared to zero when its bit position is written with one. For example writing 0x2 to CSI_PIXEL_PARSER_STATUS will clear only PPA_ILL_WD_CNT.

Offset: 0x215 | Byte Offset: 0x854 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
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.
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 does not follow the correct sequence. The correct sequence for CSI is: SF -> (SL_DATA or EF), SL_DATA -> (DATA or EL_DATA), DATA -> EL_DATA, EL_DATA -> (SL_DATA or EF), EF -> SF. Stream Errors can be caused by receiving a corrupted stream.

Bit	Reset	Description
5	X	PPA_FIFO_OVRF: FIFO Overflow, set when the FIFO that is feeding packets to PPA overflows.
4	X	PPA_PL_CRC_ERR: PayLoad CRC Error, Set when a packet that was processed by PPA had a payload CRC error.
3	X	PPA_SL_PKT_DROPPED: Short Line Packet Dropped, set when an incoming packet gets dropped because the input FIFO level reaches CSI_A_SKIP_PACKET_THRESHOLD when padding a short line.
2	X	PPA_SL_PROCESSED: Short Line Processed, Set when a line with a payload that is shorter than its packet header word count is processed by PPA.
1	X	PPA_ILL_WD_CNT: Illegal Word Count, set when a line with a word count that does not generate an integer number of pixels (Unused bytes at the end of payload) is processed by PPA.
0	X	PPA_HDR_ERR_COR: Header Error Corrected, Set when a packet that was processed by PPA has a single bit header error. This error will be detected by the headers ECC, and corrected by it if header error correction is enabled (CSI_A_HEADER_EC_ENABLE = 0). This flag will be set and the packet will be processed even if the error is not corrected.

26.16.9 CSI_CSI_SW_SENSOR_A_RESET_0

Offset: 0x216 | Byte Offset: 0x858 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSI_SENSOR_A_RESET: Reset CSI sensor A

26.16.10 CSI_INPUT_STREAM_B_CONTROL_0

CSI Input Stream B Control

Offset: 0x21b | Byte Offset: 0x86c | Read/Write: R/W | Reset: 0x00ff0004 (0bxxxxxxx011111111xxxxxxxxxx0x100)

Bit	Reset	Description
24:16	0xff	CSI_B_SKIP_PACKET_THRESHOLD: CSI-B Skip Packet Threshold. This value is compared against the internal FIFO that buffers the input streams. A packet will be skipped (discarded) if the pixel stream processor is busy (probably due to padding process of a short line) and the number of entries in the internal FIFO exceeds this threshold value. Note that each entry in the internal FIFO buffer is four bytes. To turn off this feature, set the value to its maximum value (all ones).
4	0x0	CSI_B_SKIP_PACKET_THRESHOLD_ENABLE: Enables skip packet threshold feature. 0 = DISABLE : Skip packet feature is disabled. 1 = ENABLE: Skip packet feature is enabled.
2	0x1	CSI_B_BYPASS_ALIGN: Bypass aligning CSIB lanes if valids for the lanes are out of sync by a cycle
1:0	0x0	CSI_B_DATA_LANE: CSI-B Data Lane 0= 1 data lane, 1= 2 data lanes, 2= 3 data lanes (not supported), 3= 4 data lanes (not supported) Note: 3 data lanes is not supported in Test Pattern Generation mode.

26.16.11 CSI_PIXEL_STREAM_B_CONTROL0_0

CSI Pixel Stream A Control 0

Offset: 0x21c | Byte Offset: 0x870 | Read/Write: R/W | Reset: 0xXXXX01X0 (0bxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx000)

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. 0 = PAD0S: Lines of all zeros will be used to pad out frames that are shorter than

Bit	Reset	Description
		expected height. PPB_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this field is set to PAD0S. 1 = PAD1S: Lines of all ones will be used to pad out frames that are shorter than expected height. PPB_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this field is set to PAD1S. 2 = NOPAD: Short frames will not be padded out.
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. 0 = ENABLE: Single bit errors in the header will be automatically corrected. 1 = DISABLE: 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).
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). 0 = PAD0S: short line is padded by pixel of zeros such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet may be discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 1 = PAD1S: short line is padded by pixel of ones such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet may be discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 2 = NOPAD: This option is not recommended and may cause other modules that receive CSI output stream to hang up.
21:20	X	CSI_PPB_EMBEDDED_DATA_OPTIONS: CSI Pixel Parser B Embedded Data Options. This specifies how to deal with embedded data within the specified input stream assuming that the CSI_PPB_DATA_TYPE is not embedded data and that embedded data is not already processed by another CSI pixel stream processor. 0 = DISCARD: discard (throw away) embedded data 1 = EMBEDDED: Output embedded data as 8-bpp arbitrary data stream.
19:16	X	CSI_PPB_OUTPUT_FORMAT_OPTIONS: CSI Pixel Parser B Output Format Options. This parameter specifies output data format. 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: 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.
8	0x1	CSI_PPB_WC_CHECK: CSI Pixel Parser B Data WC Check. This parameter specifies whether the wordcount is checked. 0 = DISABLE : Wordcount Check is disabled 1 = ENABLE: Wordcount Check is enabled
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. 0 = DISABLE: Data CRC Check is disabled regardless of whether there are CRC checksum at the end of the packet. 1 = ENABLE: Data CRC Check is enabled.
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. 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. Interlaced support relies on frame number and will not work in REGISTER mode. 1 = HEADER: 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

Bit	Reset	Description
		HEADER may be dangerous because the word count information in the header may be corrupted. It is recommended to always program this bit to 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. 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: Enabled - Data Identifier byte in packet header should be compared against the CSI_PPB_DATA_TYPE and the CSI_PPB_VIRTUAL_CHANNEL_ID.
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. 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: Packet header is sent. This setting should be used if the stream source is CSI Interface A or B.
2:0	0x0	CSI_PPB_STREAM_SOURCE: CSI Pixel Parser B Stream Source 0 = CSI_A: CSI Interface A 1 = CSI_B: CSI Interface B

26.16.12 CSI_PIXEL_STREAM_B_CONTROL1_0

CSI Pixel Stream B Control 1

Offset: 0x21d | Byte Offset: 0x874 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

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 OR 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. Due to the reliance on WC field in packet header, interlaced support will not work if WC is selected from REGISTER mode.

26.16.13 CSI_PIXEL_STREAM_B_GAP_0

CSI Pixel Stream B Gap

Offset: 0x21e | Byte Offset: 0x878 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PPB_FRAME_MIN_GAP: Minimum number of viclk cycles from end of frame (Video_control = EF) to start of next frame (Video_control = SF). This parameter ensures that the minimum V-blank time requirement of VI/ISP is satisfied. This field takes effect only when the frame gap of the input stream is less than the specified value.
15:0	0x0	PPB_LINE_MIN_GAP: Minimum number of viclk cycles from end of previous line (Video_control = EL_DATA) to start of next line (Video_control = SL). This parameter ensures that the minimum H-blank time requirement of VI/ISP is satisfied. This field takes effect only when the line gap of the input stream is less than the specified value.

26.16.14 CSI_PIXEL_STREAM_PPB_COMMAND_0

CSI Pixel Parser B Command

Offset: 0x21f | Byte Offset: 0x87c | Read/Write: R/W | Reset: 0x0000XXX4 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx100)

Bit	Reset	Description
15:12	X	CSI_PPB_START_MARKER_FRAME_MAX: CSI Pixel Parser B Start Marker Maximum. Start Frame is indicated when the Minimum condition above is met and the least significant four bits of the frame number are less than, or equal to, this value.
11:8	X	CSI_PPB_START_MARKER_FRAME_MIN: CSI Pixel Parser B Start Marker Minimum. Start Frame is indicated when the Maximum condition below is met and the least significant four bits of the frame number are greater than, or equal to, this value.
4	X	CSI_PPB_VSYNC_START_MARKER: CSI Pixel Parser B VSYNC Start Marker. 0 = FSPKT: Start of frame is indicated when a Frame Start short packet is received with a frame number whose least significant four bits are greater than, or equal to, CSI_PPB_START_MARKER_FRAME_MIN and less than, or equal to, CSI_PPB_START_MARKER_FRAME_MAX. When the input stream is from a CSI port, or from the host path, or from the VIP path and the data mode is PAYLOAD_ONLY, then this field may be programmed to FSPKT. 1 = VSYNC: 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.
2	0x1	CSI_PPB_SINGLE_SHOT: CSI Pixel Parser B Single Shot Mode. Software should clear it along with disabling CSI_PPB_ENABLE, once a frame is captured 0 = DISABLE 1 = ENABLE
1:0	0x0	CSI_PPB_ENABLE: CSI Pixel Parser B Enable. This parameter controls CSI Pixel Parser B to start or stop receiving data. 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: reset (disable immediately). Enabling the pixel parser does not enable the corresponding input source to receive data. If the pixel parser is enabled later than the corresponding input source, the CSI will keep rejecting the incoming stream, until it encounters a valid SF.

26.16.15 CSI_PIXEL_STREAM_B_EXPECTED_FRAME_0

CSI Pixel Stream B Expected Frame

Offset: 0x220 | Byte Offset: 0x880 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:4	X	PPB_MAX_CLOCKS: Maximum Number of viclk clock cycles between line start requests. The value in this field is in terms of 256 viclk clock cycles.
0	X	PPB_ENABLE_LINE_TIMEOUT: When set to one, enables checking of the time between start line requests from the Header Parser to CSI-PPB. A fake EF will be output by CSI-PPB if this time between line starts exceeds the value in MAX_CLOCKS_BETWEEN_LINES. Padding lines can be inserted before the fake EF, if the number of lines output, when the fake EF is generated is less than the expected frame height. The type of padding is specified using CSI_PPB_PAD_FRAME.

26.16.16 CSI_CSI_PIXEL_PARSER_B_INTERRUPT_MASK_0

CSI Pixel Parser B Interrupt Mask

Offset: 0x221 | Byte Offset: 0x884 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0xx000000000000)

Bit	Reset	Description
14	0x0	HPB_UNC_HDR_ERR_INT_MASK: Interrupt Mask for HPB_UNC_HDR_ERR. 0 = DISABLED : Do not generate an interrupt when HPB_UNC_HDR_ERR is set. 1 = ENABLED: Generate an interrupt when HPB_UNC_HDR_ERR is set.
10	0x0	PPB_SPARE_STATUS_1_INT_MASK: Interrupt Mask for PPB_SPARE_STATUS_1. 0 = DISABLED : Do not generate an interrupt when PPB_SPARE_STATUS_1 is set. 1 = ENABLED: Generate an interrupt when PPB_SPARE_STATUS_1 is set.
9	0x0	PPB_INTERFRAME_LINE_INT_MASK: Interrupt Mask for PPB_INTERFRAME_LINE. 0 = DISABLED : Do not generate an interrupt when PPB_INTERFRAME_LINE is set. 1 = ENABLED: Generate an interrupt when PPB_INTERFRAME_LINE is set.
8	0x0	PPB_EXTRA_SF_INT_MASK: Interrupt Mask for PPB_EXTRA_SF. 0 = DISABLED : Do not generate an interrupt when PPB_EXTRA_SF is set. 1 = ENABLED: Generate an interrupt when PPB_EXTRA_SF is set.
7	0x0	PPB_SHORT_FRAME_INT_MASK: Interrupt Mask for PPB_SHORT_FRAME. 0 = DISABLED : Do not generate an interrupt when PPB_SHORT_FRAME is set. 1 = ENABLED: Generate an interrupt when PPB_SHORT_FRAME is set.
6	0x0	PPB_STMERR_INT_MASK: Interrupt Mask for PPB_STMERR. 0 = DISABLED : Do not generate an interrupt when PPB_STMERR is set. 1 = ENABLED: Generate an interrupt when PPB_STMERR is set.
5	0x0	PPB_FIFO_OVRF_INT_MASK: Interrupt Mask for PPB_FIFO_OVRF. 0 = DISABLED : Do not generate an interrupt when PPB_FIFO_OVRF is set. 1 = ENABLED: Generate an interrupt when PPB_FIFO_OVRF is set.
4	0x0	PPB_PL_CRC_ERR_INT_MASK: Interrupt Mask for PPB_PL_CRC_ERR. 0 = DISABLED : Do not generate an interrupt when PPB_PL_CRC_ERR is set. 1 = ENABLED: Generate an interrupt when PPB_PL_CRC_ERR is set.
3	0x0	PPB_SL_PKT_DROPPED_INT_MASK: Interrupt Mask for PPB_SL_PKT_DROPPED. 0 = DISABLED : Do not generate an interrupt when PPB_SL_PKT_DROPPED is set. 1 = ENABLED: Generate an interrupt when PPB_SL_PKT_DROPPED is set.
2	0x0	PPB_SL_PROCESSED_INT_MASK: Interrupt Mask for PPB_SL_PROCESSED. 0 = DISABLED : Do not generate an interrupt when PPB_SL_PROCESSED is set. 1 = ENABLED: Generate an interrupt when PPB_SL_PROCESSED is set.
1	0x0	PPB_ILL_WD_CNT_INT_MASK: Interrupt Mask for PPB_ILL_WD_CNT. 0 = DISABLED : Do not generate an interrupt when PPB_ILL_WD_CNT is set. 1 = ENABLED: Generate an interrupt when PPB_ILL_WD_CNT is set.
0	0x0	PPB_HDR_ERR_COR_INT_MASK: Interrupt Mask for PPB_HDR_ERR_COR. 0 = DISABLED : Do not generate an interrupt when PPB_HDR_ERR_COR is set. 1 = ENABLED: Generate an interrupt when PPB_HDR_ERR_COR is set.

26.16.17 CSI_CSI_PIXEL_PARSER_B_STATUS_0

Pixel Parser B Status

Offset: 0x222 | Byte Offset: 0x888 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14	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.
10	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.

Bit	Reset	Description
9	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.
8	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.
7	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.
6	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.
5	X	PPB_FIFO_OVRF: FIFO Overflow, set when the FIFO that is feeding packets to PPB overflows.
4	X	PPB_PL_CRC_ERR: PayLoad CRC Error, Set when a packet that was processed by PPB had a payload CRC error.
3	X	PPB_SL_PKT_DROPPED: Short Line Packet Dropped, set when an incoming packet gets dropped because the input FIFO level reaches CSI_B_SKIP_PACKET_THRESHOLD when padding a short line.
2	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.
1	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.
0	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.

26.16.18 CSI_CSI_SW_SENSOR_B_RESET_0

Offset: 0x223 | Byte Offset: 0x88c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSI_SENSOR_B_RESET: Reset CSI sensor B

26.16.19 CSI_PHY_CIL_COMMAND_0

CSI Phy and CIL Command

Offset: 0x242 | Byte Offset: 0x908 | Read/Write: R/W | Reset: 0x00000000 (0bxx00xx00xxxxxx00xxxxxx00xxxxxx00)

Bit	Reset	Description
29:28	0x0	CSI_E_PHY_CIL_ENABLE: CSI E PHY and CIL Enable. This parameter controls CSI C PHY and CIL receiver to start or stop receiving data. disable (reset) 0 = NOP : no operation 1 = ENABLE : enable 2 = DISABLE
25:24	0x0	CSI_D_PHY_CIL_ENABLE: CSI D PHY and CIL Enable. This parameter controls CSI A PHY and CIL receiver to start or stop receiving data. 0 = NOP : no operation 1 = ENABLE : enable 2 = DISABLE
17:16	0x0	CSI_C_PHY_CIL_ENABLE: CSI C PHY and CIL Enable. This parameter controls CSI A PHY and CIL receiver to start or stop receiving data. 0 = NOP : no operation 1 = ENABLE : enable 2 = DISABLE

Bit	Reset	Description
9:8	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

26.16.20 CSI_CIL_PAD_CONFIG0_0

CIL Pad Configuration 0

Offset: 0x243 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000xxxxxxxx)

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. This 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.

26.16.21 CSI_CILA_PAD_CONFIG0_0

CIL-A Pad Configuration 0

Offset: 0x24b | Byte Offset: 0x92c | Read/Write: R/W | Reset: 0x00000007 (0b0000x000000000000000x000x0000111)

Bit	Reset	Description
31	0x0	PAD_CILA_E_TXBW: 0 = default to enable VCLAMP regulator 1 = turn off regulator and short vclamp to VDDP. will cause stress and need waiver to use
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.
23:21	0x0	PAD_CILA_LPDNADJ: Driver pull down impedance control. 00 -> 130ohm, default 01 -> 110ohm 10 -> 130ohm, same as 00 11 -> 150ohm
20:18	0x0	PAD_CILA_LPUPADJ: Driver pull up impedance control. 00 -> 130ohm, default 01 -> 110ohm 10 -> 130ohm, same as 00 11 -> 150ohm
17:16	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
15	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

Bit	Reset	Description
6:4	0x0	PAD_CILA_INADJCLK: Clock bit input delay trimmer, each tap delays 20ps
3	0x0	PAD_CILA_PDVCLAMP: Power down regulator which supplies current to serializer/deserializer logic
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

26.16.22 CSI_CILA_PAD_CONFIG1_0

CIL-A Pad Configuration 4

Offset: 0x24c | Byte Offset: 0x930 | Read/Write: R/W | Reset: 0xXXXX0000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
31:16	RO	X	PAD_CILA_SPARE_RO: Spare Read only bits for CILA Config
15:8	RW	0x0	PAD_CILA_SPARE: Spare bits for CILA Config. PAD_CILA_SPARE[15] is being used to disable the CSI-A RTL code that blocks FIFO pushes that are past the end of the line packet. 0: disabled 1: push blocking enabled
7:6	RW	0x0	PAD_CILA_PEMPD_CLK: Enable data bit HS driver pull down pre-emphasis, 00=no pre-emphasis, 11=max
5:4	RW	0x0	PAD_CILA_PEMPU_CLK: Enable data bit HS driver pull up pre-emphasis, 00=no pre-emphasis, 11=max
3:2	RW	0x0	PAD_CILA_PEMPD: Enable data bit HS driver pull down pre-emphasis, 00=no pre-emphasis, 11=max
1:0	RW	0x0	PAD_CILA_PEMPU: Enable data bit HS driver pull up pre-emphasis, 00=no pre-emphasis, 11=max

26.16.23 CSI_PHY_CILA_CONTROL0_0

CSI-A PHY and CIL Control

Offset: 0x24d | Byte Offset: 0x934 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxx000000x0000010)

Bit	Reset	Description
13:8	0x0	CILA_CLK_SETTLE: Settle time for clock lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many CSICIL clock cycles (72 MHz LP clock cycles) to wait after LP00. Hardware uses an internal delay of ~15 csicil cycles for the default value. So the programming value (0 to 15) of this field works like this: 0 = 15 cilclk cycles (default internal delay) 1 = 15 - 7 (8 cilclk cycles) 2 = 15 - 6 (9 cilclk cycles) … 7 = 15 - 1 (14 cilclk cycles) 8 = 15 - 0 (default internal delay) 9 = 15 + 1 (16 cilclk cycles) … 57 = 15 + 49 (64 cilclk cycles)
6	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 the Camera is enabled earlier than CIL, it is highly likely that the camera sends this control sequence sooner than CIL can detect it. Enabling this bit allows the CLOCK CIL to overlook the LP control sequence and step in HS Rx mode directly looking at LP00 only.
5:0	0x2	CILA_THS_SETTLE: Settle time for data lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many CSICIL clock cycles (102 MHz LP clock cycles)

Bit	Reset	Description
		to wait, after LP00, before starting to look at the data. 0xF is an illegal value for this field. 85ns + 6 * UI < (Ths-settle-programmed + 5) * csicil_clk_period < 145ns + 10 * UI

26.16.24 CSI_CSI_CIL_A_INTERRUPT_MASK_0

CSI Control and Interface Logic A Interrupt Mask

Offset: 0x24e | Byte Offset: 0x938 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0x0000x000)

Bit	Reset	Description
9	0x0	CILA_CLK_LANE_CTRL_ERR_INT_MASK: Interrupt Mask for CILA_CLK_CTRL_ERR. 0 = DISABLED : Do not generate an interrupt when CILA_CLK_CTRL_ERR is set. 1 = ENABLED: Generate an interrupt when CILA_CLK_CTRL_ERR is set.
6	0x0	CILA_ESC_DATA_REC_INT_MASK: Interrupt Mask for CILA_ESC_DATA_REC. 0 = DISABLED : Do not generate an interrupt when CILA_ESC_DATA_REC is set. 1 = ENABLED: Generate an interrupt when CILA_ESC_DATA_REC is set.
5	0x0	CILA_ESC_CMD_REC_INT_MASK: Interrupt Mask for CILA_ESC_CMD_REC. 0 = DISABLED : Do not generate an interrupt when CILA_ESC_CMD_REC is set. 1 = ENABLED: Generate an interrupt when CILA_ESC_CMD_REC is set.
4	0x0	CILA_CTRL_ERR_INT_MASK: Interrupt Mask for CILA_CTRL_ERR. 0 = DISABLED : Do not generate an interrupt when CILA_CTRL_ERR is set. 1 = ENABLED: Generate an interrupt when CILA_CTRL_ERR is set.
2	0x0	CILA_SYNC_ESC_ERR_INT_MASK: Interrupt Mask for CILA_SYNC_ESC_ERR. 0 = DISABLED : Do not generate an interrupt when CILA_SYNC_ESC_ERR is set. 1 = ENABLED: Generate an interrupt when CILA_SYNC_ESC_ERR is set.
1	0x0	CILA_SOT_MB_ERR_INT_MASK: Interrupt Mask for CILA_SOT_MB_ERR. 0 = DISABLED : Do not generate an interrupt when CILA_SOT_MB_ERR is set. 1 = ENABLED: Generate an interrupt when CILA_SOT_MB_ERR is set.
0	0x0	CILA_SOT_SB_ERR_INT_MASK: Interrupt Mask for CILA_SOT_SB_ERR. 0 = DISABLED : Do not generate an interrupt when CILA_SOT_SB_ERR is set. 1 = ENABLED: Generate an interrupt when CILA_SOT_SB_ERR is set.

26.16.25 CSI_CSI_CIL_A_STATUS_0

CSI Control and Interface Logic A Status

These status bits are cleared to zero when its bit position is written with one. For example writing 0x2 to CSI_CIL_A_STATUS will clear only CILA_SOT_MB_ERR.

Offset: 0x24f | Byte Offset: 0x93c | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
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).
2	X	CILA_SYNC_ESC_ERR: Sync Escape Error, set when CIL-A detects that the wrong (non-multiple of 8) number of bits have been received for an Escape Command, or Data Byte.
1	X	CILA_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-A detects a multi bit start of transmission byte error in one of the packet's SOT bytes. The packet will be discarded.

Bit	Reset	Description
0	X	CILA_SOT_SB_ERR: Start of Transmission Single Bit Error, set when CIL-A detects a single bit error in one of the packet's Start of Transmission bytes. The packet will be sent to the CSI-A for processing.

26.16.26 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 writing 0x2 to CSI_CIL_STATUS will clear only SOT_MB_ERR.

Offset: 0x250 | Byte Offset: 0x940 | Read/Write: RO | Reset: 0x000X00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18	X	CILA_DATA_LANE1_CTRL_ERR: Control Error, set when CIL-A detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning
17	X	CILA_DATA_LANE1_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-A detects a multi bit start of transmission byte error in one of the packet's SOT bytes on data lane-1. The packet will be discarded.
16	X	CILA_DATA_LANE1_SOT_SB_ERR: Start of Transmission Single Bit Error, set when CIL-A detects a single bit error in one of the packet's Start of Transmission bytes on data lane-1. The packet will be sent to the CSI-A for processing.
6	X	CILA_DATA_LANE0_CTRL_ERR: Control Error, set when CIL-A detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning
5	X	CILA_DATA_LANE0_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-A detects a multi bit start of transmission byte error in one of the packet's SOT bytes on data lane-0. The packet will be discarded.
4	X	CILA_DATA_LANE0_SOT_SB_ERR: Start of Transmission Single Bit Error, set when CIL-A detects a single bit error in one of the packet's Start of Transmission bytes on data lane-0. The packet will be sent to the CSI-A for processing.
0	X	CILA_CLK_LANE_CTRL_ERR: Control Error, set when CIL-A detects incorrect line state sequence on clk lane

26.16.27 CSI_CIL_A_ESCAPE_MODE_COMMAND_0

Escape Mode Command

This register is used to receive escape mode command bytes from CIL-A.

Offset: 0x251 | Byte Offset: 0x944 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
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.

26.16.28 CSI_CIL_A_ESCAPE_MODE_DATA_0

Escape Mode Data

This register is used to receive escape mode data bytes from CIL-A.

Offset: 0x252 | Byte Offset: 0x948 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
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.

26.16.29 CSI_CSICIL_SW_SENSOR_A_RESET_0

Offset: 0x253 | Byte Offset: 0x94c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSICIL_SENSOR_A_RESET: Reset CSICIL sensor A

26.16.30 CSI_CILB_PAD_CONFIG0_0

CIL-B Pad Configuration 0

Offset: 0x258 | Byte Offset: 0x960 | Read/Write: R/W | Reset: 0x00000007 (0b0000x000000000xx0000x000x0000111)

Bit	Reset	Description
31	0x0	PAD_CILB_E_TXBW: 0 = default to enable VCLAMP regulator 1 = turn off regulator and short vclamp to VDDP. will cause stress and need waiver to use
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.
23:21	0x0	PAD_CILB_LPDNADJ: Driver pull down impedance control. 00 -> 130ohm, default. 01 -> 110ohm. 10 -> 130ohm, same as 00. 11 -> 150ohm
20:18	0x0	PAD_CILB_LPUPADJ: Driver pull up impedance control. 00 -> 130ohm, default. 01 -> 110ohm. 10 -> 130ohm, same as 00. 11 -> 150ohm
15	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_PDVCLAMP: Power down regulator which supplies current to serializer/deserializer logic
2	0x1	PAD_CILB_PDIO_CLK: Power down for clock bit, including drivers, receivers and contention detectors
1:0	0x3	PAD_CILB_PDIO: Power down for each data bit, including drivers, receivers and contention detectors

26.16.31 CSI_CILB_PAD_CONFIG1_0

CIL-B Pad Configuration 4

Offset: 0x259 | Byte Offset: 0x964 | Read/Write: R/W | Reset: 0xXXXX0000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
31:16	RO	X	PAD_CILB_SPARE_RO: Spare Read only bits for CILB Config
15:8	RW	0x0	PAD_CILB_SPARE: Spare bits for CILB Config. PAD_CILB_SPARE[15] is being used to disable the CSI-B RTL code that blocks FIFO pushes that are past the end of the line packet. 0: disabled, 1: push blocking enabled
7:6	RW	0x0	PAD_CILB_PEMPD_CLK: Enable data bit HS driver pull down pre-emphasis, 00=no pre-emphasis, 11=max
5:4	RW	0x0	PAD_CILB_PEMPU_CLK: Enable data bit HS driver pull up pre-emphasis, 00=no pre-emphasis, 11=max
3:2	RW	0x0	PAD_CILB_PEMPD: Enable data bit HS driver pull down pre-emphasis, 00=no pre-emphasis, 11=max
1:0	RW	0x0	PAD_CILB_PEMPU: Enable data bit HS driver pull up pre-emphasis, 00=no pre-emphasis, 11=max

26.16.32 CSI_PHY_CILB_CONTROL0_0

CSI-B PHY and CIL Control

Offset: 0x25a | Byte Offset: 0x968 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxx000000x0000010)

Bit	Reset	Description
13:8	0x0	CILB_CLK_SETTLE: Settle time for clock lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many CSICIL clock cycles (72 MHz LP clock cycles) to wait after LP00. Hardware uses an internal delay of ~15 CSICIL cycles for default value. So the programming value (0 to 15) of this field works like this: 0 = 15 cilclk cycles (default internal delay) 1 = 15 - 7 (8 cilclk cycles) 2 = 15 - 6 (9 cilclk cycles) … 7 = 15 - 1 (14 cilclk cycles) 8 = 15 - 0 (default internal delay) 9 = 15 + 1 (16 cilclk cycles) … 57 = 15 + 49 (64 cilclk cycles)
6	0x0	CILB_BYPASS_LP_SEQ: see CILA_BYPASS_LP_SEQ above
5:0	0x2	CILB_THS_SETTLE: Settle time for data lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many CSICIL clock cycles (102 MHz LP clock cycles) to wait, after LP00, before starting to look at the data. 0xF is an illegal value for this field. 85ns + 6 * UI < (Ths-settle-programmed + 5) * csicil_clk_period < 145ns + 10 * UI

26.16.33 CSI_CSI_CIL_B_INTERRUPT_MASK_0

CSI Control and Interface Logic B Interrupt Mask

Offset: 0x25b | Byte Offset: 0x96c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0x0000x000)

Bit	Reset	Description
6	0x0	CILB_ESC_DATA_REC_INT_MASK: Interrupt Mask for CILB_ESC_DATA_REC.

Bit	Reset	Description
		0 = DISABLED : Do not generate an interrupt when CILB_ESC_DATA_REC is set. 1 = ENABLED: Generate an interrupt when CILB_ESC_DATA_REC is set.
5	0x0	CILB_ESC_CMD_REC_INT_MASK: Interrupt Mask for CILB_ESC_CMD_REC. 0 = DISABLED : Do not generate an interrupt when CILB_ESC_CMD_REC is set. 1 = ENABLED: Generate an interrupt when CILB_ESC_CMD_REC is set.
4	0x0	CILB_CTRL_ERR_INT_MASK: Interrupt Mask for CILB_CTRL_ERR. 0 = DISABLED : Do not generate an interrupt when CILB_CTRL_ERR is set. 1 = ENABLED: Generate an interrupt when CILB_CTRL_ERR is set.
2	0x0	CILB_SYNC_ESC_ERR_INT_MASK: Interrupt Mask for CILB_SYNC_ESC_ERR. 0 = DISABLED : Do not generate an interrupt when CILB_SYNC_ESC_ERR is set. 1 = ENABLED: Generate an interrupt when CILB_SYNC_ESC_ERR is set.
1	0x0	CILB_SOT_MB_ERR_INT_MASK: Interrupt Mask for CILB_SOT_MB_ERR. 0 = DISABLED : Do not generate an interrupt when CILB_SOT_MB_ERR is set. 1 = ENABLED: Generate an interrupt when CILB_SOT_MB_ERR is set.
0	0x0	CILB_SOT_SB_ERR_INT_MASK: Interrupt Mask for CILB_SOT_SB_ERR. 0 = DISABLED : Do not generate an interrupt when CILB_SOT_SB_ERR is set. 1 = ENABLED: Generate an interrupt when CILB_SOT_SB_ERR is set.

26.16.34 CSI_CSI_CIL_B_STATUS_0

CSI Control and Interface Logic B Status

Each status bit is cleared to zero when its bit position is written with one. For example writing 0x2 to CSI_CIL_B_STATUS will clear only CILB_SOT_MB_ERR.

Offset: 0x25c | Byte Offset: 0x970 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
6	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.
5	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.
4	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)..
2	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.
1	X	CILB_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-B detects a multi bit start of transmission byte error in one of the packet's SOT bytes. The packet will be discarded.
0	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.

26.16.35 CSI_CSI_CILB_STATUS_0

CSI-CILB Control and Interface Logic Status

Each status bit is cleared to zero when its bit position is written with one. For example writing 0x2 to CSI_CIL_STATUS will clear only SOT_MB_ERR.

Offset: 0x25d | Byte Offset: 0x974 | Read/Write: RO | Reset: 0x000X00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18	X	CILB_DATA_LANE1_CTRL_ERR: Control Error, set when CIL-B detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning

Bit	Reset	Description
17	X	CILB_DATA_LANE1_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-B detects a multi bit start of transmission byte error in one of the packet's SOT bytes on data lane-1. The packet will be discarded.
16	X	CILB_DATA_LANE1_SOT_SB_ERR: Start of Transmission Single Bit Error, set when CIL-A detects a single bit error in one of the packet's Start of Transmission bytes on data lane-1. The packet will be sent to the CSI-B for processing.
6	X	CILB_DATA_LANE0_CTRL_ERR: Control Error, set when CIL-B detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning
5	X	CILB_DATA_LANE0_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-B detects a multi bit start of transmission byte error in one of the packet's SOT bytes on data lane-0. The packet will be discarded.
4	X	CILB_DATA_LANE0_SOT_SB_ERR: Start of Transmission Single Bit Error, set when CIL-B detects a single bit error in one of the packet's Start of Transmission bytes on data lane-0. The packet will be sent to the CSI-B for processing.
0	X	CILB_CLK_LANE_CTRL_ERR: Control Error, set when CIL-B detects incorrect line state sequence on clk lane

26.16.36 CSI_CIL_B_ESCAPE_MODE_COMMAND_0

Escape Mode Command

This register is used to receive escape mode command bytes from CIL-B.

Offset: 0x25e | Byte Offset: 0x978 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	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. CIL-B monitors Byte Lane 0, only, for escape mode sequences. This command byte can only be assumed to be valid when CILB_ESC_CMD_REC status bit is set.

26.16.37 CSI_CIL_B_ESCAPE_MODE_DATA_0

Escape Mode Data

This register is used to receive escape mode data bytes from CIL-B.

Offset: 0x25f | Byte Offset: 0x97c | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	CILB_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-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.

26.16.38 CSI_CSICIL_SW_SENSOR_B_RESET_0

Offset: 0x260 | Byte Offset: 0x980 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSICIL_SENSOR_B_RESET: Reset CSICIL sensor B

26.16.39 CSI_CILC_PAD_CONFIG0_0

CIL-C Pad Configuration 0

Offset: 0x265 | Byte Offset: 0x994 | Read/Write: R/W | Reset: 0x00000007 (0b0000x000000000000000x000x0000111)

Bit	Reset	Description
31	0x0	PAD_CILC_E_TXBW: Increase bandwidth of output driver
30:28	0x0	PAD_CILC_SLEWDNADJ: Pull down slew rate adjust, default 000. From 000 -> 011, slew rate increases. 100 is the same as 000. From 100->111, skew rate decreases.
26:24	0x0	PAD_CILC_SLEWUPADJ: Pull up slew rate adjust, default 000. From 000 -> 011, slew rate increases. 100 is the same as 000. From 100->111, skew rate decreases.
23:21	0x0	PAD_CILC_LPDNADJ: Driver pull down impedance control 00 -> 130ohm, default. 01 -> 110ohm. 10 -> 130ohm, same as 00. 11 -> 150ohm
20:18	0x0	PAD_CILC_LPUPADJ: Driver pull up impedance control 00 -> 130ohm, default. 01 -> 110ohm. 10 -> 130ohm, same as 00. 11 -> 150ohm
17:16	0x0	PAD_CD_BK_MODE: 00: two independent 2x bricks 01: one 4x brick, received clock from partition C is used. Clock from partition D is not used 10: one 4x brick, received clock from partition D is used. Clock from partition C is not used 11: illegal
15	0x0	PAD_CILC_BANDWD_IN: Increase bandwidth of differential receiver
14:12	0x0	PAD_CILC_INADJ1: bit 1 input delay trimmer, each tap delays 20ps
10:8	0x0	PAD_CILC_INADJ0: bit 0 input delay trimmer, each tap delays 20ps
6:4	0x0	PAD_CILC_INADJCLK: Clock bit input delay trimmer, each tap delays 20ps
3	0x0	PAD_CILC_PDVCLAMP: Power down regulator which supplies current to serializer/deserializer logic
2	0x1	PAD_CILC_PDIO_CLK: Power down for clock bit, including drivers, receivers and contention detectors
1:0	0x3	PAD_CILC_PDIO: Power down for each data bit, including drivers, receivers and contention detectors

26.16.40 CSI_CILC_PAD_CONFIG1_0

CIL-C Pad Configuration 1

Offset: 0x266 | Byte Offset: 0x998 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PAD_CILC_SPARE: Spare bits for CILC Config
7:6	0x0	PAD_CILC_PEMPD_CLK: Enable data bit HS driver pull down pre-emphasis, 00=no pre-emphasis, 11=max
5:4	0x0	PAD_CILC_PEMPU_CLK: Enable data bit HS driver pull up pre-emphasis, 00=no pre-emphasis, 11=max
3:2	0x0	PAD_CILC_PEMPD: Enable data bit HS driver pull down pre-emphasis, 00=no pre-emphasis, 11=max
1:0	0x0	PAD_CILC_PEMPU: Enable data bit HS driver pull up pre-emphasis, 00=no pre-emphasis, 11=max

26.16.41 CSI_PHY_CILC_CONTROL0_0

CSI-C Phy and CIL Control

Offset: 0x267 | Byte Offset: 0x99c | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxx000000x0000010)

Bit	Reset	Description
13:8	0x0	CILC_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

Bit	Reset	Description
		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), . 57 = 15 + 49 (64 cilclk cycles)
6	0x0	CILC_BYPASS_LP_SEQ: see CILA_BYPASS_LP_SEQ above
5:0	0x2	CILC_THS_SETTLE: settle time for data lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many csicil clock cycles (102 MHz LP clock cycles) to wait, after LP00 before starting to look at the data. 0xF is an illegal value for this field 85ns + 6 * UI < (Ths-settle-programmed + 5) * csicil_clk_period < 145ns + 10 * UI

26.16.42 CSI_CSI_CIL_C_INTERRUPT_MASK_0

CSI Control and Interface Logic C Interrupt Mask

Offset: 0x268 | Byte Offset: 0x9a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0x0000x000)

Bit	Reset	Description
9	0x0	CILC_CLK_LANE_CTRL_ERR_INT_MASK: Interrupt Mask for CILC_CLK_CTRL_ERR. 0 = DISABLED : Do not generate an interrupt when CILC_CLK_CTRL_ERR is set. 1 = ENABLED: Generate an interrupt when CILC_CLK_CTRL_ERR is set.
6	0x0	CILC_ESC_DATA_REC_INT_MASK: Interrupt Mask for CILC_ESC_DATA_REC. 0 = DISABLED : Do not generate an interrupt when CILC_ESC_DATA_REC is set. 1 = ENABLED: Generate an interrupt when CILC_ESC_DATA_REC is set.
5	0x0	CILC_ESC_CMD_REC_INT_MASK: Interrupt Mask for CILC_ESC_CMD_REC. 0 = DISABLED : Do not generate an interrupt when CILC_ESC_CMD_REC is set. 1 = ENABLED: Generate an interrupt when CILC_ESC_CMD_REC is set.
4	0x0	CILC_CTRL_ERR_INT_MASK: Interrupt Mask for CILC_CTRL_ERR. 0 = DISABLED : Do not generate an interrupt when CILC_CTRL_ERR is set. 1 = ENABLED: Generate an interrupt when CILC_CTRL_ERR is set.
2	0x0	CILC_SYNC_ESC_ERR_INT_MASK: Interrupt Mask for CILC_SYNC_ESC_ERR. 0 = DISABLED : Do not generate an interrupt when CILC_SYNC_ESC_ERR is set. 1 = ENABLED: Generate an interrupt when CILC_SYNC_ESC_ERR is set.
1	0x0	CILC_SOT_MB_ERR_INT_MASK: Interrupt Mask for CILC_SOT_MB_ERR. 0 = DISABLED : Do not generate an interrupt when CILC_SOT_MB_ERR is set. 1 = ENABLED: Generate an interrupt when CILC_SOT_MB_ERR is set.
0	0x0	CILC_SOT_SB_ERR_INT_MASK: Interrupt Mask for CILC_SOT_SB_ERR. 0 = DISABLED : Do not generate an interrupt when CILC_SOT_SB_ERR is set. 1 = ENABLED: Generate an interrupt when CILC_SOT_SB_ERR is set.

26.16.43 CSI_CSI_CIL_C_STATUS_0

CSI Control and Interface Logic C 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 CILC_SOT_MB_ERR.

Offset: 0x269 | Byte Offset: 0x9a4 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
6	X	CILC_ESC_DATA_REC: Escape Mode Data Received, set when CIL-C 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 CILC_ESC_CMD_REC is set.
5	X	CILC_ESC_CMD_REC: Escape Mode Command Received, set when CIL-C receives an Escape Mode Command byte. The Command Byte can be read from bits 7-0 of ESCAPE_MODE_COMMAND.

Bit	Reset	Description
4	X	CILC_CTRL_ERR: Control Error, set when CIL-C detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning into the Escape mode or Turn Around mode (LP00).
2	X	CILC_SYNC_ESC_ERR: Sync Escape Error, set when CIL-C detects that the wrong (non-multiple of 8) number of bits have been received for an Escape Command, or Data Byte.
1	X	CILC_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-C detects a multi bit start of transmission byte error in one of the packet's SOT bytes. The packet will be discarded.
0	X	CILC_SOT_SB_ERR: Start of Transmission Single Bit Error, set when CIL-C detects a single bit error in one of the packet's Start of Transmission bytes. The packet will be sent to the CSI-C for processing.

26.16.44 CSI_CSI_CILC_STATUS_0

CSI-CILC Control and Interface Logic Status

Each status bit is cleared to zero when its bit position is written with one. For example writing 0x2 to CSI_CIL_STATUS will clear only SOT_MB_ERR.

Offset: 0x26a | Byte Offset: 0x9a8 | Read/Write: RO | Reset: 0x000X00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18	X	CILC_DATA_LANE1_CTRL_ERR: Control Error, set when CIL-C detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning
17	X	CILC_DATA_LANE1_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-C detects a multi bit start of transmission byte error in one of the packet's SOT bytes on data lane-1. The packet will be discarded.
16	X	CILC_DATA_LANE1_SOT_SB_ERR: Start of Transmission Single Bit Error, set when CIL-C detects a single bit error in one of the packet's Start of Transmission bytes on data lane-1. The packet will be sent to the CSI-C for processing.
6	X	CILC_DATA_LANE0_CTRL_ERR: Control Error, set when CIL-C detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning
5	X	CILC_DATA_LANE0_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-C detects a multi bit start of transmission byte error in one of the packet's SOT bytes on data lane-0. The packet will be discarded.
4	X	CILC_DATA_LANE0_SOT_SB_ERR: Start of Transmission Single Bit Error, set when CIL-C detects a single bit error in one of the packet's Start of Transmission bytes on data lane-0. The packet will be sent to the CSI-C for processing.
0	X	CILC_CLK_LANE_CTRL_ERR: Control Error, set when CIL-C detects incorrect line state sequence on clk lane

26.16.45 CSI_CIL_C_ESCAPE_MODE_COMMAND_0

Escape Mode Command

This register is used to receive escape mode command bytes from CIL-C.

Offset: 0x26b | Byte Offset: 0x9ac | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	CILC_ESC_CMD_BYTE: CIL-C Escape Mode Command Byte, this is the 8 bit entry command that was received by CIL-C, during the last escape Mode sequence. CIL-C monitors Byte Lane 0, only, for escape mode sequences. This command byte can only be assumed to be valid when CILC_ESC_CMD_REC status bit is set.

26.16.46 CSI_CIL_C_ESCAPE_MODE_DATA_0

Escape Mode Data

This register is used to receive escape mode data bytes from CIL-C.

Offset: 0x26c | Byte Offset: 0x9b0 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	CILC_ESC_DATA_BYTE: CIL-C Escape Mode Data Byte, when read this field returns the last Escape Mode Data byte that was received by CIL-C. 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, CILC_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.

26.16.47 CSI_CSICIL_SW_SENSOR_C_RESET_0

Offset: 0x26d | Byte Offset: 0x9b4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSICIL_SENSOR_C_RESET: Reset CSICIL sensor C

26.16.48 CSI_CILD_PAD_CONFIG0_0

CIL-D Pad Configuration 0

Offset: 0x272 | Byte Offset: 0x9c8 | Read/Write: R/W | Reset: 0x00000007 (0b0000x000000000xx0000x000x0000111)

Bit	Reset	Description
31	0x0	PAD_CILD_E_TXBW: Increase bandwidth of output driver
30:28	0x0	PAD_CILD_SLEWDNADJ: Pull down slew rate adjust, default 000. From 000 -> 011, slew rate increases. 100 is the same as 000. From 100->111, skew rate decreases.
26:24	0x0	PAD_CILD_SLEWUPADJ: Pull up slew rate adjust, default 000. From 000 -> 011, slew rate increases. 100 is the same as 000. From 100->111, skew rate decreases.
23:21	0x0	PAD_CILD_LPDNADJ: Driver pull down impedance control. 00 -> 130ohm, default. 01 -> 110ohm. 10 -> 130ohm, same as 00. 11 -> 150ohm
20:18	0x0	PAD_CILD_LPUPADJ: Driver pull up impedance control 00 -> 130ohm, default. 01 -> 110ohm. 10 -> 130ohm, same as 00. 11 -> 150ohm
15	0x0	PAD_CILD_BANDWD_IN: Increase bandwidth of differential receiver
14:12	0x0	PAD_CILD_INADJ1: bit 1 input delay trimmer, each tap delays 20ps
10:8	0x0	PAD_CILD_INADJ0: bit 0 input delay trimmer, each tap delays 20ps
6:4	0x0	PAD_CILD_INADJCLK: Clock bit input delay trimmer, each tap delays 20ps
3	0x0	PAD_CILD_PDVCLAMP: Power down regulator which supplies current to serializer/deserializer logic
2	0x1	PAD_CILD_PDIO_CLK: Power down for clock bit, including drivers, receivers and contention detectors
1:0	0x3	PAD_CILD_PDIO: Power down for each data bit, including drivers, receivers and contention detectors

26.16.49 CSI_CILD_PAD_CONFIG1_0

CIL-D Pad Configuration 1

Offset: 0x273 | Byte Offset: 0x9cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PAD_CILD_SPARE: Spare bits for CILD Config
7:6	0x0	PAD_CILD_PEMPD_CLK: Enable data bit HS driver pull down pre-emphasis, 00=no pre-emphasis, 11=max
5:4	0x0	PAD_CILD_PEMPU_CLK: Enable data bit HS driver pull up pre-emphasis, 00=no pre-emphasis, 11=max
3:2	0x0	PAD_CILD_PEMPD: Enable data bit HS driver pull down pre-emphasis, 00=no pre-emphasis, 11=max
1:0	0x0	PAD_CILD_PEMPU: Enable data bit HS driver pull up pre-emphasis, 00=no pre-emphasis, 11=max

26.16.50 CSI_PHY_CILD_CONTROL0_0

CSI-D Phy and CIL Control

Offset: 0x274 | Byte Offset: 0x9d0 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxx000000x0000010)

Bit	Reset	Description
13:8	0x0	CILD_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), . 57 = 15 + 49 (64 cilclk cycles)
6	0x0	CILD_BYPASS_LP_SEQ: see CILA_BYPASS_LP_SEQ above
5:0	0x2	CILD_THS_SETTLE: settle time for data lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many csicil clock cycles (102 MHz lp clock cycles) to wait, after LP00, before starting to look at the data. 0xF is an illegal value for this field 85ns + 6 * UI < (Ths-settle-programmed + 5) * csicil_clk_period < 145ns + 10 * UI

26.16.51 CSI_CSI_CIL_D_INTERRUPT_MASK_0

CSI Control and Interface Logic D Interrupt Mask

Offset: 0x275 | Byte Offset: 0x9d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0x0000x000)

Bit	Reset	Description
6	0x0	CILD_ESC_DATA_REC_INT_MASK: Interrupt Mask for CILD_ESC_DATA_REC. 0 = DISABLED : Do not generate an interrupt when CILD_ESC_DATA_REC is set. 1 = ENABLED: Generate an interrupt when CILD_ESC_DATA_REC is set.
5	0x0	CILD_ESC_CMD_REC_INT_MASK: Interrupt Mask for CILD_ESC_CMD_REC. 0 = DISABLED : Do not generate an interrupt when CILD_ESC_CMD_REC is set. 1 = ENABLED: Generate an interrupt when CILD_ESC_CMD_REC is set.
4	0x0	CILD_CTRL_ERR_INT_MASK: Interrupt Mask for CILD_CTRL_ERR. 0 = DISABLED : Do not generate an interrupt when CILD_CTRL_ERR is set. 1 = ENABLED: Generate an interrupt when CILD_CTRL_ERR is set.
2	0x0	CILD_SYNC_ESC_ERR_INT_MASK: Interrupt Mask for CILD_SYNC_ESC_ERR. 0 = DISABLED : Do not generate an interrupt when CILD_SYNC_ESC_ERR is set. 1 = ENABLED: Generate an interrupt when CILD_SYNC_ESC_ERR is set.
1	0x0	CILD_SOT_MB_ERR_INT_MASK: Interrupt Mask for CILD_SOT_MB_ERR. 0 = DISABLED : Do not generate an interrupt when CILD_SOT_MB_ERR is set. 1 = ENABLED: Generate an interrupt when CILD_SOT_MB_ERR is set.

Bit	Reset	Description
0	0x0	CILD_SOT_SB_ERR_INT_MASK: Interrupt Mask for CILD_SOT_SB_ERR. 0 = DISABLED : Do not generate an interrupt when CILD_SOT_SB_ERR is set. 1 = ENABLED: Generate an interrupt when CILD_SOT_SB_ERR is set.

26.16.52 CSI_CSI_CIL_D_STATUS_0

CSI Control and Interface Logic D Status

Offset: 0x276 | Byte Offset: 0x9d8 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
6	X	CILD_ESC_DATA_REC: Escape Mode Data Received, set when CIL-D 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 CILD_ESC_CMD_REC is set.
5	X	CILD_ESC_CMD_REC: Escape Mode Command Received, set when CIL-D receives an Escape Mode Command byte. The Command Byte can be read from bits 7-0 of ESCAPE_MODE_COMMAND.
4	X	CILD_CTRL_ERR: Control Error, set when CIL-D detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning into the Escape mode or Turn Around mode (LP00).
2	X	CILD_SYNC_ESC_ERR: Sync Escape Error, set when CIL-D detects that the wrong (non-multiple of 8) number of bits have been received for an Escape Command, or Data Byte.
1	X	CILD_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-D detects a multi bit start of transmission byte error in one of the packet's SOT bytes. The packet will be discarded.
0	X	CILD_SOT_SB_ERR: Start of Transmission Single Bit Error, set when CIL-D detects a single bit error in one of the packet's Start of Transmission bytes. The packet will be sent to the CSI-D for processing.

26.16.53 CSI_CSI_CILD_STATUS_0

CSI-CILD Control and Interface Logic Status

Each status bit is cleared to zero when its bit position is written with one. For example writing 0x2 to CSI_CIL_STATUS will clear only SOT_MB_ERR.

Offset: 0x277 | Read/Write: RO | Reset: 0x000X00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18	X	CILD_DATA_LANE1_CTRL_ERR: Control Error, set when CIL-D detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning
17	X	CILD_DATA_LANE1_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-D detects a multi bit start of transmission byte error in one of the packet's SOT bytes on data lane-1. The packet will be discarded.
16	X	CILD_DATA_LANE1_SOT_SB_ERR: Start of Transmission Single Bit Error, set when CIL-D detects a single bit error in one of the packet's Start of Transmission bytes on data lane-1. The packet will be sent to the CSI-D for processing.
6	X	CILD_DATA_LANE0_CTRL_ERR: Control Error, set when CIL-D detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning
5	X	CILD_DATA_LANE0_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-D detects a multi bit start of transmission byte error in one of the packet's SOT bytes on data lane-0. The packet will be discarded.
4	X	CILD_DATA_LANE0_SOT_SB_ERR: Start of Transmission Single Bit Error, set when CIL-D detects a single bit error in one of the packet's Start of Transmission bytes on data lane-0. The packet will be sent to the CSI-D for processing.
0	X	CILD_CLK_LANE_CTRL_ERR: Control Error, set when CIL-D detects incorrect line state sequence on clk lane

26.16.54 CSI_CIL_D_ESCAPE_MODE_COMMAND_0

Escape Mode Command

This register is used to receive escape mode command bytes from CIL-D.

Offset: 0x27b | Byte Offset: 0x9ec | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	CILD_ESC_CMD_BYTE: CIL-D Escape Mode Command Byte, this is the 8-bit entry command that was received, by CIL-D, during the last escape Mode sequence. CIL-D monitors Byte Lane 0, only, for escape mode sequences. This command byte can only be assumed to be valid when CILD_ESC_CMD_REC status bit is set.

26.16.55 CSI_CIL_D_ESCAPE_MODE_DATA_0

Escape Mode Data

This register is used to receive escape mode data bytes from CIL-D.

Offset: 0x27c | Byte Offset: 0x9f0 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	CILD_ESC_DATA_BYTE: CIL-D Escape Mode Data Byte, when read this field returns the last Escape Mode Data byte that was received by CIL-D. 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 Specifications Low Power Data Transmission. This field is only valid when the status bit, CILD_ESC_DATA_REC, is set, and will be overwritten by the next Escape Mode data byte if not read before the next byte come in.

26.16.56 CSI_CSICIL_SW_SENSOR_D_RESET_0

Offset: 0x27d | Byte Offset: 0x9f4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSICIL_SENSOR_D_RESET: Reset CSICIL sensor D

26.16.57 CSI_CILE_PAD_CONFIG0_0

CIL-E Pad Configuration 0

Offset: 0x282 | Byte Offset: 0xa08 | Read/Write: R/W | Reset: 0x00000005 (0b0000x000000000xx0xxxx000x00001x1)

Bit	Reset	Description
31	0x0	PAD_CILE_E_TXBW: Increase bandwidth of output driver
30:28	0x0	PAD_CILE_SLEWDNADJ: Pull down slew rate adjust, default 000. From 000 -> 011, slew rate increases. 100 is the same as 000. From 100->111, skew rate decreases.
26:24	0x0	PAD_CILE_SLEWUPADJ: Pull up slew rate adjust, default 000. From 000 -> 011, slew rate increases. 100 is the same as 000. From 100->111, skew rate decreases.
23:21	0x0	PAD_CILE_LPDNADJ: Driver pull down impedance control. 00 -> 130ohm, default. 01 -> 110ohm. 10 -> 130ohm, same as 00. 11 -> 150ohm
20:18	0x0	PAD_CILE_LPUPADJ: Driver pull up impedance control. 00 -> 130ohm, default. 01 -> 110ohm. 10 -> 130ohm, same as 00. 11 -> 150ohm
15	0x0	PAD_CILE_BANDWD_IN: Increase bandwidth of differential receiver
10:8	0x0	PAD_CILE_INADJ0: bit 0 input delay trimmer, each tap delays 20ps

Bit	Reset	Description
6:4	0x0	PAD_CILE_INADJCLK: Clock bit input delay trimmer, each tap delays 20ps
3	0x0	PAD_CILE_PDVCLAMP: Power down regulator which supplies current to serializer/deserializer logic
2	0x1	PAD_CILE_PDIO_CLK: Power down for clock bit, including drivers, receivers and contention detectors
0	0x1	PAD_CILE_PDIO: Power down for each data bit, including drivers, receivers and contention detectors

26.16.58 CSI_CILE_PAD_CONFIG1_0

CIL-B Pad Configuration 4

Offset: 0x283 | Byte Offset: 0xa0c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:6	0x0	PAD_CILE_PEMPD_CLK: Enable data bit HS driver pull down pre-emphasis, 00=no pre-emphasis, 11=max
5:4	0x0	PAD_CILE_PEMPU_CLK: Enable data bit HS driver pull up pre-emphasis, 00=no pre-emphasis, 11=max
3:2	0x0	PAD_CILE_PEMPD: Enable data bit HS driver pull down pre-emphasis, 00=no pre-emphasis, 11=max
1:0	0x0	PAD_CILE_PEMPU: Enable data bit HS driver pull up pre-emphasis, 00=no pre-emphasis, 11=max

26.16.59 CSI_PHY_CILE_CONTROL0_0

CSI-E PHY and CIL Control

Offset: 0x284 | Byte Offset: 0xa10 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxx000000x0000010)

Bit	Reset	Description
13:8	0x0	CILE_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), . 57 = 15 + 49 (64 cilclk cycles)
6	0x0	CILE_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.
5:0	0x2	CILE_THS_SETTLE: settle time for data lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many CSICIL clock cycles (102 MHz lp clock cycles) to wait, after LP00, before starting to look at the data. 0xF is an illegal value for this field 85ns + 6 * UI < (Ths-settle-programmed + 5) * csicil_clk_period < 145ns + 10 * UI

26.16.60 CSI_CSI_CIL_E_INTERRUPT_MASK_0

CSI Control and Interface Logic Interrupt Mask

Offset: 0x285 | Byte Offset: 0xa14 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0x0000x000)

Bit	Reset	Description
9	0x0	CILE_CLK_LANE_CTRL_ERR_INT_MASK: Interrupt Mask for CILE_CLK_CTRL_ERR. 0 = DISABLED : Do not generate an interrupt when CILE_CLK_CTRL_ERR is set. 1 = ENABLED: Generate an interrupt when CILE_CLK_CTRL_ERR is set.
7	0x0	CILE_SPARE_STATUS_1_INT_MASK: Interrupt Mask for CILE_SPARE_STATUS_1. 0 = DISABLED : Do not generate an interrupt when CILE_SPARE_STATUS_1 is set.

Bit	Reset	Description
		1 = ENABLED: Generate an interrupt when CILE_SPARE_STATUS_1 is set.
6	0x0	CILE_ESC_DATA_REC_INT_MASK: Interrupt Mask for CILE_ESC_DATA_REC. 0 = DISABLED : Do not generate an interrupt when CILE_ESC_DATA_REC is set. 1 = ENABLED: Generate an interrupt when CILE_ESC_DATA_REC is set.
5	0x0	CILE_ESC_CMD_REC_INT_MASK: Interrupt Mask for CILE_ESC_CMD_REC. 0 = DISABLED : Do not generate an interrupt when CILE_ESC_CMD_REC is set. 1 = ENABLED: Generate an interrupt when CILE_ESC_CMD_REC is set.
4	0x0	CILE_CTRL_ERR_INT_MASK: Interrupt Mask for CILE_CTRL_ERR. 0 = DISABLED : Do not generate an interrupt when CILE_CTRL_ERR is set. 1 = ENABLED: Generate an interrupt when CILE_CTRL_ERR is set.
2	0x0	CILE_SYNC_ESC_ERR_INT_MASK: Interrupt Mask for CILE_SYNC_ESC_ERR. 0 = DISABLED : Do not generate an interrupt when CILE_SYNC_ESC_ERR is set. 1 = ENABLED: Generate an interrupt when CILE_SYNC_ESC_ERR is set.
1	0x0	CILE_SOT_MB_ERR_INT_MASK: Interrupt Mask for CILE_SOT_MB_ERR. 0 = DISABLED : Do not generate an interrupt when CILE_SOT_MB_ERR is set. 1 = ENABLED: Generate an interrupt when CILE_SOT_MB_ERR is set.
0	0x0	CILE_SOT_SB_ERR_INT_MASK: Interrupt Mask for CILE_SOT_SB_ERR. 0 = DISABLED : Do not generate an interrupt when CILE_SOT_SB_ERR is set. 1 = ENABLED: Generate an interrupt when CILE_SOT_SB_ERR is set.

26.16.61 CSI_CSI_CIL_E_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 writing 0x1 to CSI_CIL_STATUS will clear only CILE_SOT_SB_ERR.

Offset: 0x286 | Byte Offset: 0xa18 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8	X	CILE_CLK_LANE_CTRL_ERR: Control Error, set when CIL-E detects incorrect line state sequence on clk lane
6	X	CILE_ESC_DATA_REC: Escape Mode Data Received, set when CIL-E 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 CILE_ESC_CMD_REC is set.
5	X	CILE_ESC_CMD_REC: Escape Mode Command Received, set when CIL-E receives an Escape Mode Command byte. The Command Byte can be read from bits 7-0 of ESCAPE_MODE_COMMAND.
4	X	CILE_CTRL_ERR: Control Error, set when CIL-E detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning into the Escape mode or Turn Around mode (LP00).
2	X	CILE_SYNC_ESC_ERR: Sync Escape Error, set when CIL-E detects that the wrong (non-multiple of 8) number of bits have been received for an Escape Command, or Data Byte.
1	X	CILE_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-C detects a multi bit start of transmission byte error in one of the packet's SOT bytes. The packet will be discarded.
0	X	CILE_SOT_SB_ERR: Start of Transmission Single Bit Error, set when CIL-E detects a single bit error in one of the packet's Start of Transmission bytes. The packet will be sent to the CSI-C for processing.

26.16.62 CSI_CIL_E_ESCAPE_MODE_COMMAND_0

Escape Mode Command

This register is used to receive escape mode command bytes from CIL-E.

Offset: 0x287 | Byte Offset: 0xa1c | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	CILE_ESC_CMD_BYTE: CIL-E Escape Mode Command Byte, this is the 8 bit entry command that was received by CIL-E, during the last escape Mode sequence. CIL-E monitors Byte Lane 0, only, for escape mode sequences. This command byte can only be assumed to be valid when CILE_ESC_CMD_REC status bit is set.

26.16.63 CSI_CIL_E_ESCAPE_MODE_DATA_0

Escape Mode Data

This register is used to receive escape mode data bytes from CIL-E.

Offset: 0x288 | Byte Offset: 0xa20 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	CILE_ESC_DATA_BYTE: CIL-E Escape Mode Data Byte, when read this field returns the last Escape Mode Data byte that was received by CIL-E. 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, CILE_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.

26.16.64 CSI_CSICIL_SW_SENSOR_E_RESET_0

Offset: 0x289 | Byte Offset: 0xa24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSICIL_SENSOR_E_RESET: Reset CSICIL sensor E

26.16.65 CSI_PATTERN_GENERATOR_CTRL_A_0

Offset: 0x29a | Byte Offset: 0xa68 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:2	0x0	PG_MODE_A: Mode for Sensor A 0 = DIRECT 1 = PATCH 2 = RSVD
1	0x0	PG_AUTO_INC_A: Automatic phase increment mode for sensor A
0	0x0	PG_ENABLE_A: Enable Pattern Generator for sensor A

26.16.66 CSI_PG_BLANK_A_0

Offset: 0x29b | Byte Offset: 0xa6c | Read/Write: R/W | Reset: 0x00080008 (0b00000000000010000000000000001000)

Bit	Reset	Description
31:16	0x8	PG_VBLANK_A: Vertical Blanking for PG
15:0	0x8	PG_HBLANK_A: Horizontal Blanking for PG

26.16.67 CSI_PG_PHASE_A_0

Offset: 0x29c | Byte Offset: 0xa70 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00000000000000)

Bit	Reset	Description
13:0	0x0	PG_PHASE_A: Initial Phase

26.16.68 CSI_PG_RED_FREQ_A_0

Offset: 0x29d | Byte Offset: 0xa74 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_RED_VERT_INIT_FREQ_A: Initial vertical frequency
13:0	0x0	PG_RED_HOR_INIT_FREQ_A: Initial horizontal frequency

26.16.69 CSI_PG_RED_FREQ_RATE_A_0

Offset: 0x29e | Byte Offset: 0xa78 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_RED_VERT_FREQ_RATE_A: Rate of change of vertical frequency
7:0	0x0	PG_RED_HOR_FREQ_RATE_A: Rate of change of horizontal frequency

26.16.70 CSI_PG_GREEN_FREQ_A_0

Offset: 0x29f | Byte Offset: 0xa7c | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_GREEN_VERT_INIT_FREQ_A: Initial vertical frequency
13:0	0x0	PG_GREEN_HOR_INIT_FREQ_A: Initial horizontal frequency

26.16.71 CSI_PG_GREEN_FREQ_RATE_A_0

Offset: 0x2a0 | Byte Offset: 0xa80 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_GREEN_VERT_FREQ_RATE_A: Rate of change of vertical frequency
7:0	0x0	PG_GREEN_HOR_FREQ_RATE_A: Rate of change of horizontal frequency

26.16.72 CSI_PG_BLUE_FREQ_A_0

Offset: 0x2a1 | Byte Offset: 0xa84 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_BLUE_VERT_INIT_FREQ_A: Initial vertical frequency
13:0	0x0	PG_BLUE_HOR_INIT_FREQ_A: Initial horizontal frequency

26.16.73 CSI_PG_BLUE_FREQ_RATE_A_0

Offset: 0x2a2 | Byte Offset: 0xa88 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_BLUE_VERT_FREQ_RATE_A: Rate of change of vertical frequency
7:0	0x0	PG_BLUE_HOR_FREQ_RATE_A: Rate of change of horizontal frequency

26.16.74 CSI_PATTERN_GENERATOR_CTRL_B_0

Offset: 0x2a7 | Byte Offset: 0xa9c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:2	0x0	PG_MODE_B: Mode for Sensor B 0 = DIRECT 1 = PATCH 2 = RSVD
1	0x0	PG_AUTO_INC_B: Automatic phase increment mode for sensor B
0	0x0	PG_ENABLE_B: Enable Pattern Generator for sensor B

26.16.75 CSI_PG_BLANK_B_0

Offset: 0x2a8 | Byte Offset: 0xaa0 | Read/Write: R/W | Reset: 0x00080008 (0b00000000000010000000000000001000)

Bit	Reset	Description
31:16	0x8	PG_VBLANK_B: Vertical Blanking for PG
15:0	0x8	PG_HBLANK_B: Horizontal Blanking for PG

26.16.76 CSI_PG_PHASE_B_0

Offset: 0x2a9 | Byte Offset: 0xaa4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00000000000000)

Bit	Reset	Description
13:0	0x0	PG_PHASE_B: Initial Phase

26.16.77 CSI_PG_RED_FREQ_B_0

Offset: 0x2aa | Byte Offset: 0xaa8 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_RED_VERT_INIT_FREQ_B: Initial vertical frequency
13:0	0x0	PG_RED_HOR_INIT_FREQ_B: Initial horizontal frequency

26.16.78 CSI_PG_RED_FREQ_RATE_B_0

Offset: 0x2ab | Byte Offset: 0xaac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_RED_VERT_FREQ_RATE_B: Rate of change of vertical frequency
7:0	0x0	PG_RED_HOR_FREQ_RATE_B: Rate of change of horizontal frequency

26.16.79 CSI_PG_GREEN_FREQ_B_0

Offset: 0x2ac | Byte Offset: 0xab0 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_GREEN_VERT_INIT_FREQ_B: Initial vertical frequency
13:0	0x0	PG_GREEN_HOR_INIT_FREQ_B: Initial horizontal frequency

26.16.80 CSI_PG_GREEN_FREQ_RATE_B_0

Offset: 0x2ad | Byte Offset: 0xab4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_GREEN_VERT_FREQ_RATE_B: Rate of change of vertical frequency
7:0	0x0	PG_GREEN_HOR_FREQ_RATE_B: Rate of change of horizontal frequency

26.16.81 CSI_PG_BLUE_FREQ_B_0

Offset: 0x2ae | Byte Offset: 0xab8 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_BLUE_VERT_INIT_FREQ_B: Initial vertical frequency
13:0	0x0	PG_BLUE_HOR_INIT_FREQ_B: Initial horizontal frequency

26.16.82 CSI_PG_BLUE_FREQ_RATE_B_0

Offset: 0x2af | Byte Offset: 0xabc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_BLUE_VERT_FREQ_RATE_B: Rate of change of vertical frequency
7:0	0x0	PG_BLUE_HOR_FREQ_RATE_B: Rate of change of horizontal frequency

26.16.83 CSI_DPCM_CTRL_A_0

Offset: 0x2b4 | Byte Offset: 0xad0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxxxxx0)

Bit	Reset	Description
11:8	0x0	DPCM_COMPRESSION_RATIO_A: DPCM A compression ratio 0 : BYPASS 1 : 10_8_10 2 : 10_7_10 3 : 10_6_10 4 : 12_8_12 5 : 12_7_12 6 : 12_6_12 7 : 14_10_14 8 : 14_8_14
0	0x0	DPCM_PREDICTOR_A: DPCM A predictor 0: predictor1 1: predictor2

26.16.84 CSI_DPCM_CTRL_B_0

Offset: 0x2b5 | Byte Offset: 0xad4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxxxxx0)

Bit	Reset	Description
11:8	0x0	DPCM_COMPRESSION_RATIO_B: DPCM A compression ratio 0 : BYPASS 1 : 10_8_10 2 : 10_7_10 3 : 10_6_10 4 : 12_8_12 5 : 12_7_12 6 : 12_6_12 7 : 14_10_14 8 : 14_8_14
0	0x0	DPCM_PREDICTOR_B: DPCM A predictor 0: predictor1 1: predictor2

26.16.85 CSI_STALL_COUNTER_0

Offset: 0x2ba | Byte Offset: 0xae8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	STALL_SENSOR_B_COUNT
7:0	0x0	STALL_SENSOR_A_COUNT: Number of cycles to stall sensor A after every EOF

26.16.86 CSI_CSI_READONLY_STATUS_0

CSI Read Only Status

This register is used to return CSI read only status.

Offset: 0x2bb | Byte Offset: 0xaec | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	CSI_PPB_ACTIVE: One only when Pixel Parser B is capturing frame data.
0	X	CSI_PPA_ACTIVE: One only when Pixel Parser A is capturing frame data.

26.16.87 CSI_CSI_SW_STATUS_RESET_0

Offset: 0x2bc | Byte Offset: 0xaf0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSI_STATUS_RESET: Reset CSI status and dbgcnt registers

26.16.88 CSI_CLKEN_OVERRIDE_0

Second-level clock enable override register

This can override the 2nd level clock enables in case of malfunction. Only exposed to software when needed.

Offset: 0x2bd | Byte Offset: 0xaf4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx00000xx00xxx00xxx00x00)

Bit	Reset	Description
21	CLK_GATED	CSI_CILE_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
20	CLK_GATED	CSI_CILD_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
19	CLK_GATED	CSI_CILC_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
18	CLK_GATED	CSI_CILB_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
17	CLK_GATED	CSI_CILA_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
14	CLK_GATED	CSI_PPB_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
13	CLK_GATED	CSI_PPA_CLKEN_OVR:

Bit	Reset	Description
		0 = CLK_GATED 1 = CLK_ALWAYS_ON
9	CLK_GATED	CSI_HPB_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
8	CLK_GATED	CSI_HPA_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
4	CLK_GATED	CSI_FB_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
3	CLK_GATED	CSI_FA_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
1	CLK_GATED	CSI_DBG_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
0	CLK_GATED	CSI_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON

26.16.89 CSI_DEBUG_CONTROL_0

Debug Control

Offset: 0x2be | Byte Offset: 0xaf8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	R/W	Reset	Description
31	RO	X	DBG_CNT_ROLLED_2: Set when dbg_cnt_2 is incremented past max count, cleared when clr_dbg_cnt_2 is written with a value of 1.
30:24	RW	X	DBG_CNT_SEL_2: Debug Count Select 2, this field selects what will be counted by debug counter 2. Encodings 00 to 31 select the set signal for one of the CSI_PIXEL_PARSER_STATUS register bits. In this case the select encoding of this field is the same as the bit position, in CSI_PIXEL_PARSER_STATUS, of the status bit whose set signal will be used to increment DBG_CNT_0. Encodings 32 to 63 select the set signal for one of the CSI_CIL_STATUS status bits. The least significant 5 bits of this select field give the bit position, in CSI_CIL_STATUS, of the status bit whose set signal pulses will be counted by dbg_cnt_0. Selections for encodings 64 to 127 are given below: 64 - PPA Line packets processed 65 - PPA short packets processed 66 - Total packets processed by PPA 67 - PPA Frame Starts Outputted 68 - PPA Frame Ends Outputted 69 - Reserved encoding 70 - PPB Line packets processed 71 - PPB short packets processed 72 - Total packets processed by PPB 73 - PPB Frame Starts Outputted 74 - PPB Frame Ends Outputted 75 - Reserved encoding 76 - HPA Headers Parsed 77 - HPA Headers Parsed with no ECC Errors 78 - HPB Headers Parsed 79 - HPB Headers Parsed with no ECC Errors 80 - HPV Headers Parsed 81 - HPV Headers Parsed with no ECC Errors 82 - HPH Headers Parsed 83 - HPH Headers Parsed with no ECC Errors 84 - 32 bit words read from vi2csi_host_data 85 to 127 - Reserved encodings
23	RO	X	DBG_CNT_ROLLED_1: Set when dbg_cnt_1 is incremented past max count, cleared when clr_dbg_cnt_1 is written with a value of 1.
22:16	RW	X	DBG_CNT_SEL_1: Debug Count Select 1, this field selects what will be counted by debug counter 1. Encodings 00 to 31 select the set signal for one of the CSI_PIXEL_PARSER_STATUS register bits. In this case the select encoding of this field is the same as the bit position, in CSI_PIXEL_PARSER_STATUS, of the status bit whose set signal will be used to increment DBG_CNT_0. Encodings 32 to 63 select the set signal for one of the CSI_CIL_STATUS status bits. The least significant 5 bits of this select field give the bit position, in CSI_CIL_STATUS, of the status bit whose set signal pulses will be counted by dbg_cnt_0. Selections for encodings 64 to 127 are given below: 64 - PPA Line packets processed 65 - PPA short packets processed 66 - Total packets processed by PPA 67 - PPA Frame Starts Outputted 68 - PPA Frame Ends Outputted 69 - Reserved encoding 70 - PPB Line packets processed 71 - PPB short packets processed 72 - Total packets processed by PPB 73 - PPB Frame Starts Outputted 74 - PPB

Bit	R/W	Reset	Description
			Frame Ends Outputted 75 - Reserved encoding 76 - HPA Headers Parsed 77 - HPA Headers Parsed with no ECC Errors 78 - HPB Headers Parsed 79 - HPB Headers Parsed with no ECC Errors 80 - HPV Headers Parsed 81 - HPV Headers Parsed with no ECC Errors 82 - HPH Headers Parsed 83 - HPH Headers Parsed with no ECC Errors 84 - 32 bit words read from vi2csi_host_data 85 to 127 - Reserved encodings
15	RO	X	DBG_CNT_ROLLED_0: Set when dbg_cnt_0 is incremented past max count, cleared when clr_dbg_cnt_0 is written with a value of 1.
14:8	RW	X	DBG_CNT_SEL_0: Debug Count Select 0, this field selects what will be counted by debug counter 0. Encodings 00 to 31 select the set signal for one of the CSI_PIXEL_PARSER_STATUS register bits. In this case the select encoding of this field is the same as the bit position, in CSI_PIXEL_PARSER_STATUS, of the status bit whose set signal will be used to increment DBG_CNT_0. Encodings 32 to 63 select the set signal for one of the CSI_CIL_STATUS status bits. The least significant 5 bits of this select field give the bit position, in CSI_CIL_STATUS, of the status bit whose set signal pulses will be counted by dbg_cnt_0. Selections for encodings 64 to 127 are given below: 64 - PPA Line packets processed 65 - PPA short packets processed 66 - Total packets processed by PPA 67 - PPA Frame Starts Outputted 68 - PPA Frame Ends Outputted 69 - Reserved encoding 70 - PPB Line packets processed 71 - PPB short packets processed 72 - Total packets processed by PPB 73 - PPB Frame Starts Outputted 74 - PPB Frame Ends Outputted 75 - Reserved encoding 76 - HPA Headers Parsed 77 - HPA Headers Parsed with no ECC Errors 78 - HPB Headers Parsed 79 - HPB Headers Parsed with no ECC Errors 80 - HPV Headers Parsed 81 - HPV Headers Parsed with no ECC Errors 82 - HPH Headers Parsed 83 - HPH Headers Parsed with no ECC Errors 84 - 32 bit words read from vi2csi_host_data 85 to 127 - Reserved encodings
7	RO	X	CLR_DBG_CNT_2: Clear Debug Counter 2, write a one to this bit to clear debug counter 2 and dbg_cnt_rolled_2.
6	RO	X	CLR_DBG_CNT_1: Clear Debug Counter 1, write a one to this bit to clear debug counter 1 and dbg_cnt_rolled_1.
5	RO	X	CLR_DBG_CNT_0: Clear Debug Counter 0, write a one to this bit to clear debug counter 0 and dbg_cnt_rolled_0.
2	RO	X	CSIB_DBG_SF: When CSI-B is operating in a "Header Not Sent mode", writing a 1 to this bit indicates start frame (SF) or end frame (EF) control code. After the pixel parser is enabled, writing a 1 to this bit will start frame capture and send start frame (SF) control code. Writing a 1 to this bit again will stop frame capture and send end frame (EF) control code. "Header Not Sent mode" can be used as a debug mode to capture what the sensor is sending without interpreting the packets. Writing a 1 to this bit continually will generate SF and EF control codes. Note that a wait for MISC_CSI_PPB_FRAME_END syncpt is needed between an EF trigger for the current frame and an SF trigger for the next frame.
1	RO	X	CSIA_DBG_SF: When CSI-A is operating in a "Header Not Sent mode", writing a 1 to this bit indicates start frame (SF) or end frame (EF) control code. After the pixel parser is enabled, writing a 1 to this bit will start frame capture and send start frame (SF) control code. Writing a 1 to this bit again will stop frame capture and send end frame (EF) control code. "Header Not Sent mode" can be used as a debug mode to capture what the sensor is sending without interpreting the packets. Writing a 1 to this bit continually will generate SF and EF control codes. Note that a wait for MISC_CSI_PPA_FRAME_END syncpt is needed between an EF trigger for the current frame and an SF trigger for the next frame.
0	RW	0x0	DEBUG_EN: Debug Enable Second level CSI Debug clock is enabled. Debug counters 2, 1 & 0 are powered up. 0 = DISABLED : Debug counters 2, 1 & 0 are powered down. Second level CSI Debug clock is disabled. 1 = ENABLED

26.16.90 CSI_DEBUG_COUNTER_0_0

Debug Counter 0

This register can be used to count error conditions or packets processed.

Offset: 0x2bf | Byte Offset: 0xafc | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DBG_CNT_0: When read returns the value of debug counter 0.

26.16.91 CSI_DEBUG_COUNTER_1_0

Debug Counter 1

This register can be used to count error conditions or packets processed.

Offset: 0x2c0 | Byte Offset: 0xb00 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DBG_CNT_1: When read returns the value of debug counter 1.

26.16.92 CSI_DEBUG_COUNTER_2_0

Debug Counter 2

This register can be used to count error conditions or packets processed.

Offset: 0x2c1 | Byte Offset: 0xb04 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DBG_CNT_2: When read returns the value of debug counter 2

[THIS PAGE INTENTIONALLY LEFT BLANK]

27.0 MIPI D-PHY CALIBRATION FOR CSI AND DSI

This section describes the MIPI D-PHY Calibration registers. This information is provided to aid in system debug and diagnostics. It is expected that NVIDIA® supplied drivers will normally be used to program these.

MIPI D-PHY blocks are used for both the DSI display interface and the CSI camera interface. Both interfaces are controlled by these registers.

27.1 MIPI-CAL Registers

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

27.1.1 MIPI_CAL_MIPI_CAL_CTRL_0

Calibration Control Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x2a000000 (0bxx101010xxxxxxxxxxxxxxxxxxxx0xx00)

Bit	R/W	Reset	Description
29:26	RW	0xa	MIPI_CAL_NOISE_FLT: The DRIVRY and TERMRY signals coming from MIPI pads are utilized by the Calibration state machine for pad calibration. The drivry/termry comes from a noisy analog source, which might have some glitches. The filter in calibsm is sensitive to these noises. If the calibration done status does not show up, the sensitivity of the filter can be changed through these bits. Ideally, this has to be programmed in a range from 10 to 15. When MIPI_CAL_PRESCALE = 2'b00, this needs to be programmed between 2 to 5.
25:24	RW	0x2	MIPI_CAL_PRESCALE: Auto-Cal calibration step prescale: 00: when calibration step is 0.1 µs 01: when calibration step is 0.5 µs 10: when calibration step is 1.0 µs (default) 11: when calibration step is 1.5 µs This keeps the MIPI bias calibration step between 0.1-1.5 µs.
4	RW	CLK_GATED	MIPI_CAL_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
1	RW	0x0	MIPI_CAL_AUTOCAL_EN: When set, calibration is triggered periodically. The Period is set using MIPI_CAL_AUTOCAL_CTRL0 register
0	RO	0x0	MIPI_CAL_STARTCAL: Writing a one to this bit starts the Calibration State machine. This bit must be set even if both overrides are set in order to latch in the override value.

27.1.2 MIPI_CAL_MIPI_CAL_AUTOCAL_CTRL0_0

Offset: 0x4 | Read/Write: R/W | Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:0	-1	MIPI_CAL_AUTOCAL_PERIOD: Auto-calibration period in number of APB sys clk cycles.

27.1.3 MIPI_CAL_CIL_MIPI_CAL_STATUS_0

CIL MIPI Calibrate Status

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

Bit	Reset	Description
29	X	MIPI_AUTO_CAL_DONE_DSIB: MIPI Auto Calibrate done for DSIB. Set when the auto calibrate sequence for DSI pad bricks is done. Write 1-to-clear.
28	X	MIPI_AUTO_CAL_DONE_DSIA: MIPI Auto Calibrate done for DSI. Set when the auto calibrate sequence for DSI pad bricks is done. Write 1-to-clear.
24	X	MIPI_AUTO_CAL_DONE_CSIE: MIPI Auto Calibrate done for CSI, set when the auto calibrate sequence for CSI pad bricks is done. Write 1-to-clear.
23	X	MIPI_AUTO_CAL_DONE_CSID: MIPI Auto Calibrate done for CSI. Set when the auto calibrate sequence for CSI pad bricks is done. Write 1-to-clear.
22	X	MIPI_AUTO_CAL_DONE_CSIC: Debug bit. MIPI Auto Calibrate done for CSI. Set when the auto calibrate sequence for CSI pad bricks is done. Write 1-to-clear.
21	X	MIPI_AUTO_CAL_DONE_CSIB: MIPI Auto Calibrate done for CSI. Set when the auto calibrate sequence for CSI pad bricks is done. Write 1-to-clear.
20	X	MIPI_AUTO_CAL_DONE_CSIA: MIPI Auto Calibrate done for CSI. Set when the auto calibrate sequence for CSI pad bricks is done. Write 1-to-clear.
16	X	MIPI_AUTO_CAL_DONE: MIPI Auto Calibrate done. Set when the auto-calibrate sequence for MIPI pad bricks is done. Write 1-to-clear.
15:12	X	MIPI_CAL_DRIV_DN_ADJ: Driver Pull-down calibration code generated by MIPI auto calibrate. Valid only after the auto-calibrate sequence has completed (MIPI_CAL_ACTIVE == 0).
11:8	X	MIPI_CAL_DRIV_UP_ADJ: Driver Pull up calibration code generated by MIPI auto calibrate. Valid only after auto calibrate sequence has completed (MIPI_CAL_ACTIVE == 0).
7:4	X	MIPI_CAL_TERMADJ: Termination code generated by MIPI auto calibrate. Valid only after the auto-calibrate sequence has completed (MIPI_CAL_ACTIVE == 0).
0	X	MIPI_CAL_ACTIVE: One when auto calibrate is active.

27.1.4 MIPI_CAL_CIL_MIPI_CAL_STATUS_2_0

MIPI CLK Calibration Status 2

Offset: 0xc | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
4	X	MIPI_CLK_AUTO_CAL_DONE_DSIB: MIPI CLK Auto Calibrate done for DSIB, set when the auto calibrate sequence for DSI pad bricks is done. Write 1-to-clear.
3	X	MIPI_CLK_AUTO_CAL_DONE_DSIA: MIPI CLK Auto Calibrate done for DSIA, set when the auto calibrate sequence for DSI pad bricks is done. Write 1-to-clear.
2	X	MIPI_CLK_AUTO_CAL_DONE_CSIE: MIPI CLK Auto Calibrate done for CSIE, set when the auto calibrate sequence for CSI pad bricks is done. Write 1-to-clear.
1	X	MIPI_CLK_AUTO_CAL_DONE_CSID: MIPI CLK Auto Calibrate done for CSID, set when the auto calibrate sequence for CSI pad bricks is done. Write 1-to-clear.
0	X	MIPI_CLK_AUTO_CAL_DONE_CSIC: MIPI CLK Auto Calibrate done for CSIC, set when the auto calibrate sequence for CSI pad bricks is done. Write 1-to-clear.

27.1.5 MIPI_CAL_CILA_MIPI_CAL_CONFIG_0

Calibration Settings for CIL-A MIPI Pads

Offset: 0x14 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxx100000xxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEA: When 0 (normal operation), use TERMOS/HSPUOS/HSPDOS as an offset to the Calibration State machine setting for channel A TERMADJ/HSPUADJ/HSPDADJ values to the MIPI pads. When 1, use the register values above as the actual value going to channel A TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads.
21	0x1	MIPI_CAL_SELA: Select the CSIA pads for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSA: 2's complement offset for HSPDADJ going to channel A (currently negative offsets are not supported)
12:8	0x0	MIPI_CAL_HSPUOSA: 2's complement offset for HSPUADJ going to channel A (currently negative offsets are not supported)
4:0	0x0	MIPI_CAL_TERMOSA: 2's complement offset for TERMADJ going to channel A (currently negative offsets are not supported)

27.1.6 MIPI_CAL_CILB_MIPI_CAL_CONFIG_0

Calibration Settings for CIL-B MIPI Pads

Offset: 0x18 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxx100000xxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEB: When 0 (normal operation), use TERMOS/HSPUOS/HSPDOS as an offset to the Calibration State machine setting for channel B TERMADJ/HSPUADJ/HSPDADJ values to the MIPI pads. When 1, use the register values above as the actual value going to channel B TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads. Writing a one to MIPI_CAL_STARTCAL starts the Calibration State machine.
21	0x1	MIPI_CAL_SELB: Select the CSIB pads for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSB: 2's complement offset for HSPDADJ going to channel B (currently negative offsets are not supported)
12:8	0x0	MIPI_CAL_HSPUOSB: 2's complement offset for HSPUADJ going to channel B (currently negative offsets are not supported)
4:0	0x0	MIPI_CAL_TERMOSB: 2's complement offset for TERMADJ going to channel B (currently negative offsets are not supported)

27.1.7 MIPI_CAL_CILC_MIPI_CAL_CONFIG_0

Calibration Settings for CIL-C MIPI Pads

Offset: 0x1c | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxx100000xxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEC: When 0 (normal operation), use TERMOS/HSPUOS/HSPDOS as an offset to the Calibration State machine setting for channel C TERMADJ/HSPUADJ/HSPDADJ values to the MIPI pads. When 1, use the register values above as the actual value going to channel C TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads. Writing a one to MIPI_CAL_STARTCAL starts the Calibration State machine.
21	0x1	MIPI_CAL_SELC: Select the CSIC pads for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSC: 2's complement offset for HSPDADJ going to channel C (currently negative offsets are not supported)

Bit	Reset	Description
12:8	0x0	MIPI_CAL_HSPUOSC: 2's complement offset for HSPUADJ going to channel C (currently negative offsets are not supported)
4:0	0x0	MIPI_CAL_TERMOSC: 2's complement offset for TERMADJ going to channel C (currently negative offsets are not supported)

27.1.8 MIPI_CAL_CILD_MIPI_CAL_CONFIG_0

Calibration Settings for CIL-D MIPI Pads

Offset: 0x20 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxxx100000xxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDED: When 0 (normal operation), use TERMOS/HSPUOS/HSPDOS as an offset to the Calibration State machine setting for channel D TERMADJ/HSPUADJ/HSPDADJ values to the MIPI pads. When 1, use the register values above as the actual value going to channel D TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads. Writing a one to MIPI_CAL_STARTCAL starts the Calibration State machine.
21	0x1	MIPI_CAL_SELD: Select the CSID PADS for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSD: 2's complement offset for HSPDADJ going to channel D (currently negative offsets are not supported)
12:8	0x0	MIPI_CAL_HSPUOSD: 2's complement offset for HSPUADJ going to channel D (currently negative offsets are not supported)
4:0	0x0	MIPI_CAL_TERMOSD: 2's complement offset for TERMADJ going to channel D (currently negative offsets are not supported)

27.1.9 MIPI_CAL_CILE_MIPI_CAL_CONFIG_0

Calibration Settings for CIL-E MIPI Pads

Offset: 0x24 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxxx100000xxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEE: When 0 (normal operation), use TERMOS/HSPUOS/HSPDOS as an offset to the Calibration State machine setting for channel E TERMADJ/HSPUADJ/HSPDADJ values to the MIPI pads. When 1, use the register values above as the actual value going to channel E TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads. Writing a one to MIPI_CAL_STARTCAL starts the Calibration State machine.
21	0x1	MIPI_CAL_SELE: Select the CSIE pads for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSE: 2's complement offset for HSPDADJ going to channel E (currently negative offsets are not supported)
12:8	0x0	MIPI_CAL_HSPUOSE: 2's complement offset for HSPUADJ going to channel E (currently negative offsets are not supported)
4:0	0x0	MIPI_CAL_TERMOSE: 2's complement offset for TERMADJ going to channel E (currently negative offsets are not supported)

27.1.10 MIPI_CAL_DSIA_MIPI_CAL_CONFIG_0

Calibration Settings for DSIA MIPI Pad

Offset: 0x38 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxx100000xxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEDSIA: When 0 (normal operation), use TERMOS/HSPUOS/HSPDOS as an offset to the Calibration State machine setting for TERMADJ/HSPUADJ/HSPDADJ values to the MIPI pads. When 1, use the register values above as the actual value going to TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads. Writing a one to MIPI_CAL_STARTCAL starts the Calibration State machine.
21	0x1	MIPI_CAL_SELDSIA: Select the DSI pads for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSDSIA: 2's complement offset for HSPDADJ (currently negative offsets are not supported)
12:8	0x0	MIPI_CAL_HSPUOSDSIA: 2's complement offset for HSPUADJ (currently negative offsets are not supported)
4:0	0x0	MIPI_CAL_TERMOSDSIA: 2's complement offset for TERMADJ (currently negative offsets are not supported)

27.1.11 MIPI_CAL_DSIB_MIPI_CAL_CONFIG_0

Calibration Settings for DSIB MIPI Pad

Offset: 0x3c | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxx100000xxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEDSIB: When 0 (normal operation), use TERMOS/HSPUOS/HSPDOS as an offset to the Calibration State machine setting for TERMADJ/HSPUADJ/HSPDADJ values to the MIPI pads. When 1, use the register values above as the actual value going to TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads. Writing a one to MIPI_CAL_STARTCAL starts the Calibration State machine.
21	0x1	MIPI_CAL_SELDSIB: Select the DSI pads for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSDSIB: 2's complement offset for HSPDADJ (currently negative offsets are not supported)
12:8	0x0	MIPI_CAL_HSPUOSDSIB: 2's complement offset for HSPUADJ (currently negative offsets are not supported)
4:0	0x0	MIPI_CAL_TERMOSDSIB: 2's complement offset for TERMADJ (currently negative offsets are not supported)

27.1.12 MIPI_CAL_MIPI_BIAS_PAD_CFG0_0

MIPI Bias Pad Configuration Register

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	MIPI_BIAS_PAD_PDVCLAMP: 1=Power down regulator which supplies current to pre-driver logic.
0	0x0	MIPI_BIAS_PAD_E_VCLAMP_REF: 1=Enable VCLAMP reference voltage

27.1.13 MIPI_CAL_MIPI_BIAS_PAD_CFG1_0

MIPI Bias Pad Configuration Register

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx000xxxxx000xxxxx000xxxxx000)

Bit	Reset	Description
26:24	0x0	PAD_TEST_SEL: Controls which signal to be routed to TEST_OUT 000=VAUXP 001=RUP 010=RDN 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

27.1.14 MIPI_CAL_MIPI_BIAS_PAD_CFG2_0

MIPI Bias Pad Configuration Register 2

Offset: 0x60 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxx00000000000x000x010)

Bit	Reset	Description
18:16	0x0	PAD_VCLAMP_LEVEL: VCLAMP level adjustment
15:8	0x0	PAD_SPARE: Spare bit for MIPI Bias Config
6:4	0x0	PAD_VAUXP_LEVEL: VAUXP level adjustment 00 = no adjustment, default 01 = 105% 10 = 110% 11 = 115% 100 = no adjustment 101 = 95% 110 = 90% 111 = 85%
2	0x0	PAD_E_TXBW: This bit should not be used. Set it to 0.
1	0x1	PAD_PDVREG: Power down voltage regulator, 1=power down
0	0x0	PAD_VBYPASS: Bypass bang gap voltage reference

27.1.15 MIPI_CAL_DSIA_MIPI_CAL_CONFIG_2_0

Calibration settings for DSIA pad Channel A

Offset: 0x64 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxxx1xxxxxxxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_CLKOVERIDEDSIA: When 0 (normal operation), use the above registers as an offset to the Calibration State machine setting for channel A HSCLKPUADJ/HSCLKPDADJ values to the MIPI Pads. When 1, use the register values above as the actual values going to channel A HSCLKPUADJ/HSCLKPDADJ on the MIPI Pads.
21	0x1	MIPI_CAL_CLKSELDSIA: Select the DSIA PAD Channel A for clock lane auto calibration.
12:8	0x0	MIPI_CAL_HSCLKPDOSDSIA: 2's complement offset for HSCLKPDADJ going to Channel B
4:0	0x0	MIPI_CAL_HSCLKPUOSDSIA: 2's complement offset for HSCLKPUADJ going to Channel A

27.1.16 MIPI_CAL_DSIB_MIPI_CAL_CONFIG_2_0

Calibration settings for DSIA pad Channel B

Offset: 0x68 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxxx1xxxxxxxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_CLKOVERIDEDSIB: When 0 (normal operation), use the above registers as an offset to the Calibration State machine setting for channel B HSCLKPUADJ/HSCLKPDADJ values to the MIPI Pads. When 1, use the register values above as the actual values going to channel B HSCLKPUADJ/HSCLKPDADJ on the MIPI Pads.
21	0x1	MIPI_CAL_CLKSELDSIB: Select the DSIA PAD Channel B for clock lane auto calibration.
12:8	0x0	MIPI_CAL_HSCLKPDOSDSIB: 2's complement offset for HSCLKPDADJ
4:0	0x0	MIPI_CAL_HSCLKPUOSDSIB: 2's complement offset for HSCLKPUADJ

27.1.17 MIPI_CAL_CILC_MIPI_CAL_CONFIG_2_0

Calibration settings for DSIB/CSICD pad

Offset: 0x6c | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxxx1xxxxxxxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_CLKOVERIDEC: When 0 (normal operation), use the above registers as an offset to the Calibration State machine setting for channel A HSCLKPUADJ/HSCLKPDADJ values to the MIPI Pads. When 1, use the register values above as the actual values going to channel A HSCLKPUADJ/HSCLKPDADJ on the MIPI Pads.
21	0x1	MIPI_CAL_CLKSELC: Select the DSIB PAD Channel A for clock lane auto calibration.
12:8	0x0	MIPI_CAL_HSCLKPDOSC: 2's complement offset for HSCLKPDADJ going to channel A
4:0	0x0	MIPI_CAL_HSCLKPUOSC: 2's complement offset for HSCLKPUADJ going to channel A

27.1.18 MIPI_CAL_CILD_MIPI_CAL_CONFIG_2_0

Calibration settings for DSIB/CSICD pad

Offset: 0x70 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxxx1xxxxxxxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_CLKOVERIDED: When 0 (normal operation), use the above registers as an offset to the Calibration State machine setting for channel B HSCLKPUADJ/HSCLKPDADJ values to the MIPI Pads. When 1, use the register values above as the actual values going to channel B HSCLKPUADJ/HSCLKPDADJ on the MIPI Pads.
21	0x1	MIPI_CAL_CLKSELD: Select the DSIB PAD Channel B for clock lane auto calibration.
12:8	0x0	MIPI_CAL_HSCLKPDOSD: 2's complement offset for HSCLKPDADJ going to channel B
4:0	0x0	MIPI_CAL_HSCLKPUOSD: 2's complement offset for HSCLKPUADJ going to channel B

27.1.19 MIPI_CAL_CSIE_MIPI_CAL_CONFIG_2_0

Calibration settings for CSIE MIPI pad

Offset: 0x74 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxxx1xxxxxxxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_CLKOVERIDEE: When 0 (normal operation), use the above registers as an offset to the

Bit	Reset	Description
		Calibration State machine setting for channel A HSCLKPUADJ/HSCLKPDADJ values to the MIPI Pads. When 1, use the register values above as the actual values going to channel A HSCLKPUADJ/HSCLKPDADJ on the MIPI Pads.
21	0x1	MIPI_CAL_CLKSELE: Select the CSIE PAD for clock lane auto calibration.
12:8	0x0	MIPI_CAL_HSCLKPDOSE: 2's complement offset for HSCLKPDADJ
4:0	0x0	MIPI_CAL_HSCLKPUOSE: 2's complement offset for HSCLKPUADJ

28.0 VIDEO INPUT (VI)

The Video Input 2 unit (VI2) receives data from the CSI unit and provides for its presentation to either system memory or the dedicated ISP execution resources. The VI2 provides for formatting RGB, YCbCr, and raw Bayer data in support of a number of user models. These models include single and multi-camera systems, which may use up to two image/video streams obtained from MIPI compliant CMOS sensor camera modules. In addition to packing of image data, several advanced use cases are supported with processing of the raw Bayer data. These user modes involve the capture of high-resolution still images while simultaneously maintaining a lower resolution preview or video capture through the ISP, and the capture of stereo video and still images.

This section also describes the VI registers; however, it is not intended to be a programming guide to VI, as it is expected that NVIDIA supplied drivers will always be used with it. However, the register interface is documented to aid with understanding those drivers.

28.1 Functionality

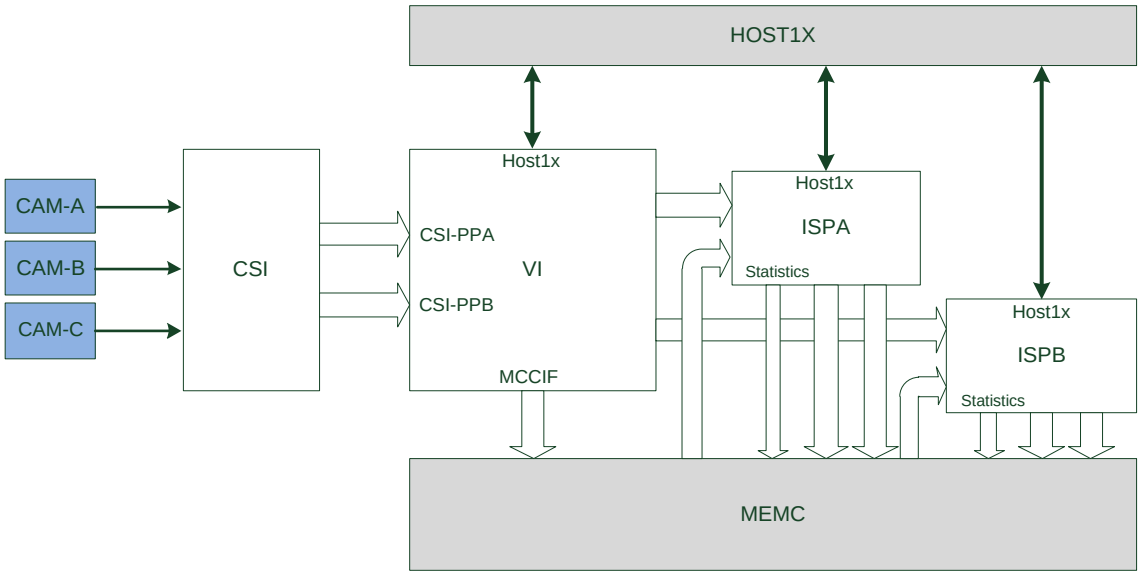
28.1.1 Camera Subsystem

The VI2 unit interfaces with four major functional units within the Tegra® K1 camera subsystem in addition to the Host1x, which is used for control programming:

- Camera Serial Interface (CSI) – Provides for the interface to MIPI compliant CMOS sensors and formats the pixel data for presentation to the VI2.
- Image Signal Processors A and B (ISPA and ISPB) – Execution resource providing image processing in support of various camera use cases and raw Bayer processing.
- Memory Controller (MC) – Interface between the VI2 and ISP and memory resources.

The main paths for the flow of image data for the VI2 are from the CSI → VI2 → ISPA/ISPB and from the CSI → VI2 → MC. There are also several additional paths for data flow within the camera's subsystem, which includes paths from the VI2 to MC and the MC to the ISP. The ability to present data to system memory allows customer-specific algorithms, such as blur reduction, chromatic aberration correction, and high ISO noise processing, to be inserted into the ISP pipeline. An added benefit is the ability to divide input data into more manageable widths when processing high-resolution still images. The high-level block diagram for the Tegra K1 camera subsystem is illustrated in the following figure.

Figure 93: Tegra K1 Camera Subsystem



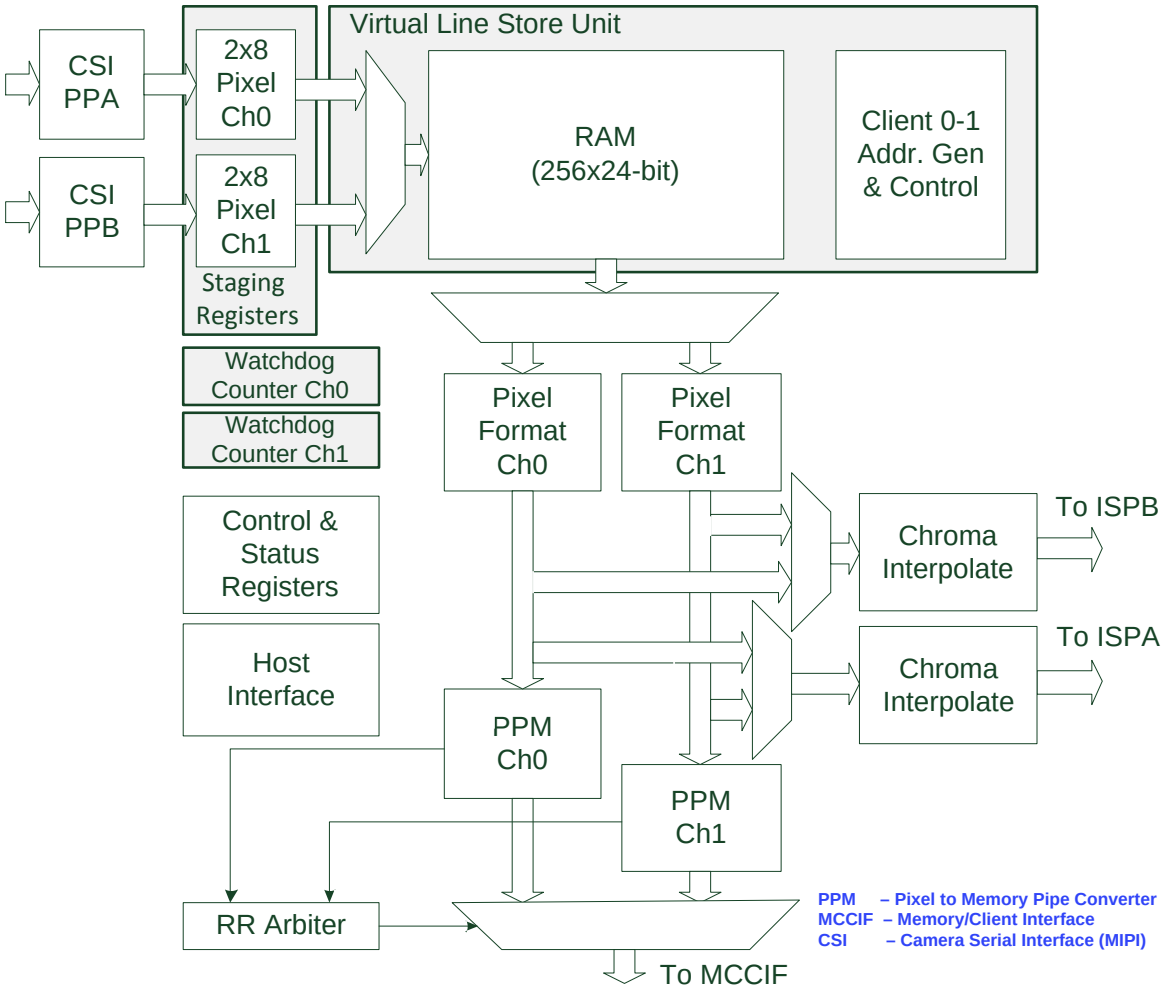
The support for multiple paths to and from memory within the camera subsystem is useful for very wide surfaces, such as still capture, where the alternative would be to have very large on-chip line stores. Because there are two ISP resources, two data streams from separate sensors or camera modules can be provided for immediate processing in support of dual and stereo camera capture models. In addition to the basic multi-camera user modes, the output path from the VI2 to memory allows the maintenance of a Zero Shutter Lag (ZSL) circular buffer for high-resolution capture while the ISP is processing the preview stream for display.

The Host1x interface handles the flow of the programmed control state. The CSI registers are in the VI2 unit and are accessed through the Host interface of the VI2. Refer to the CSI section in this document for more information.

28.1.2 Block Diagram

The VI2 unit accepts pixel data from an external source, packages the received data, and presents the data to either system memory or the ISP resources directly. The pixel data received can be in a variety of formats including Raw, RGB, and YCbCr. The high-level block diagram for the VI2 is shown in the following figure.

Figure 94: VI High-Level Block Diagram

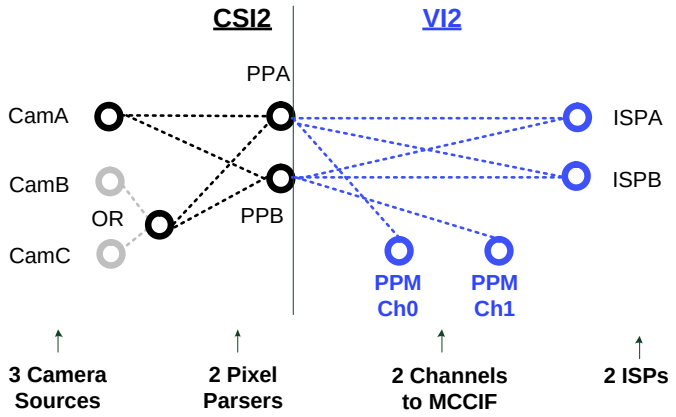


The VI2 provides a dual-stream capture mode in which two different resolution images are available to the system to support the various camera use cases. A high-resolution image can be captured and saved to memory while simultaneously presenting the image for processing through the ISP in support of scene preview and composition. The dual camera modes include support for stereo still and video capture, and the mixed format capture in support of video conference or scene annotation.

28.1.3 Single and Dual Stream Datapaths

The Tegra K1 camera subsystem provides considerable flexibility in the support of various single and dual camera user models. This flexibility is manifested as a number of programmable configurations that allow the image data to be directed to system memory and ISP execution resources. The datapaths for the camera subsystem are illustrated in the following figure.

Figure 95: Single and Dual Stream Datapaths through the CSI2 and VI2



The CSI2 unit provides for connection of up to three cameras in the system with the limitation that only two can be active simultaneously. The CSI2 can direct the data streams to either of the pixel parsers, PPA or PPB. In addition to supporting a dual stream capture from two sources, the CSI2 supports a broadcast mode in which a single camera data stream can be broadcast to both PPA and PPB simultaneously. In the broadcast mode the same data stream can be presented to the ISP or memory resources in a number of different ways. A summary of the various single camera configurations is provided in the following table.

Table 110: Datapath Options through VI2 for Single Camera User Models

Datapath Active for Single Camera Source	Comments
ISPA	Single stream capture through PPA or PPB
ISPB	Single stream capture through PPA or PPB
ISPA & ISPB	Single stream broadcast through both PPA and PPB
ISPA & ISPB & (PPM0 or PPM1)	Single stream broadcast through both PPA and PPB; single stream to MEM through PPM Channel 0 or PPM Channel 1.
ISPA & ISPB & PPM0 & PPM1	Single stream broadcast through both PPA and PPB; both ISPA and ISPB and both PPM Channel 0 and PPM Channel 1 active.
ISPA & (PPM0 or PPM1)	Single stream capture through PPA or PPB; single stream to MEM through PPM Ch0 or PPM Ch1.
ISPA & PPM0 & PPM1	Single stream broadcast through both PPA and PPB; PPM Ch0 and PPM Ch1 active.
ISPB & (PPM0 or PPM1)	Single stream capture through PPA or PPB; single stream to MEM through PPM Channel 0 or PPM Channel 1.
PPM0 or PPM1	Single stream capture through PPA or PPB; single stream to MEM through PPM Channel 0 or PPM Channel 1.
PPM0 & PPM1	Single stream broadcast through both PPA and PPB; both ISPA and ISPB and both PPM Channel 0 and PPM Channel 1 active.

The dual camera use cases have the limitation that CAMA is always active with either CAMB or CAMC. This limitation results because CAMB and CAMC share the same lane merging logic in the CSI allowing only one to be available at a time. The dual camera user models have a similar level of flexibility as the single camera except that variations due to a broadcast mode are not valid.

Table 111: Datapath Options through VI2 for Two Camera User Models

Datapaths Active for Dual Camera Sources		Comments
CAMA	CAMB or CAMC	
ISPA	ISPB	Dual Stream Capture through PPA and PPB
ISPB	ISPA	Dual Stream Capture through PPA and PPB
ISPA & (PPM0 or PPM1)	ISPB	Data streaming to memory through PPA is routed through PPM Channel 0 and if PPB it is routed thru PPM Channel 1.
ISPB & (PPM0 or PPM1)	ISPA	Data streaming to memory through PPA is routed through PPM Channel 0 and if PPB it is routed through PPM Channel 1.
ISPA	ISPB & (PPM0 or PPM1	Data streaming to memory through PPA is routed through PPM Channel 0 and if PPB it is routed through PPM Channel 1.
ISPB	ISPA & (PPM0 or PPM1)	Data streaming to memory through PPA is routed through PPM Channel 0 and if PPB it is routed through PPM Channel 1.
ISPA & (PPM0 or PPM1)	ISPB & (PPM0 or PPM1)	Data streaming to memory through PPA is routed through PPM Channel 0 and if PPB it is routed through PPM Channel 1.
ISPB & (PPM0 or PPM1)	ISPA & (PPM0 or PPM1)	Data streaming to memory through PPA is routed through PPM Channel 0 and if PPB it is routed through PPM Channel 1.

28.1.4 Host1x Interface

The Host1x interface is the primary method for communicating control state from the CPU to the VI2. It is responsible for the register write and read requests. The VI2 has total of two channels for Host1x interface. This provides flexibility to the software to simultaneously capture images from two cameras asynchronous to each other. A single channel is not bound to any camera and software can map any channel to any camera.

The Host1x interface of the VI2 is used to configure the registers for both VI2 and CSI units. The rest of the section also describes some of the CSI related logic.

The registers can be categorized in two categories: configuration registers and operational registers. The configuration registers should remain constant for the life of the camera operation (until the camera unit is re-enabled between application switching). The operational registers are required to process a given frame of image.

In order to process the images with minimum VBlank, the operational registers need to be double-buffered to speed up the transition between the operational parameters between the two frames.

28.1.4.1 State Updates

The operation registers are initially assembled in the assembly state. The registers are copied to the active state when the pipe is idle or immediately after startup. In all other cases, they are copied to the active state when the update command is received and when the next EOF is received. The following table describes the VI state update trigger.

Table 112: Single Shot Register Definition

Register	Field	Bits	Reset	Description
VI_CSI_[0..1]_SINGLE_SHOT	CAPTURE	0	0	Software sets this bit to request transition of the previously assembled assembly state to the active state and schedule single shot capture. The actual

Register	Field	Bits	Reset	Description
				update of the operational register is gated by the next EOF. Register read back returns the current capture status 0: No capture pending. 1: Capture pending
VI_CSI_[0..1]_SINGLE_SHOT_UPDATE	CAPTURE_GOOD_FRAME	0	1	This bit controls when the update command should be processed. 1'b0: Transition to update when the next EOF is received, irrespective of the state of previous frame. **Note1:** Useful for debugging and/or bring up when the sensor/VI configuration could be incorrect. 1'b1: Transition to next operational state when the next EOF is received AND the previous frame was the correct frame based on the current operational state. **Note2:** Any mismatch will result in the frame getting dropped and the CSI rearming with unchanged config. **Note3**: After exiting the reset state, the previous frame with the correct state should be set.

In the normal functional mode of operation, the EOF updates the pending assembly state to the active state only if the previous frame was successfully captured (or this is the first frame out of reset). If for any reason the previous frame was detected to be an errored frame, on the next SOF, the VI2 and CSI unit continue to work on the previous operational state configuration and do not update to the next set of operational parameters. An additional configuration register controls the behavior of VI2 unit.

28.1.4.2 Sync Points

Sync points (syncpts) are a means of synchronizing software programmed control state and hardware operations, without the need for register polling or interrupt service routines. Host1X contains a number of counters. These counters are referenced through an ***index***. Software can issue two kinds of syncpt-related methods:

- INCR_SYNCPT ***<condition> <index>***
- WAIT_SYNCPT ***<index> <threshold>***

INCR_SYNCPT tells the hardware that when ***<condition>*** is met, the hardware should send the value of ***<index>*** to the Host1X on the RtnIndx bus. On receiving the index return, Host1X will increment the counter pointed to by ***<index>***.

WAIT_SYNCPT is a Host1X method which tells Host1X to cease processing commands and register writes for that unit until the value of the counter pointed to by ***<index>*** reaches or exceeds the value of ***<threshold>***.

Since software can be made to wait for the counter to reach a certain value and hardware controls the incrementing of the counter, the two can be kept synchronized, ensuring that control state is metered out at the appropriate times by Host1X. In

order to support the single shot and shadow register behaviors, sync point conditions 0x03 thru 0x0F are buffered, allowing up to two sync points for each of these conditions to be programmed. The following table describes the VI2 sync point conditions.

Table 113: Sync Point Conditions

Condition	Value	FIFO Depth	Description
IMMEDIATE	0x00	1	Predefined Immediate condition. The VI2 returns this syncpt immediately.
OP_DONE	0x01	1	Predefined Operation Done Condition
RD_DONE	0x02	1	Predefined Read Done condition. The VI2 treats this as IMMEDIATE because CSI/VI do not have any read ports.
REG_WR_SAFE	0x03	2	Predefined Register Write Safe condition. Technically software should know when it is safe to write to the functional registers because it knows when CSI/VI is idle. The VI2 instead generates this syncpt when there are no pending captures and pending OP*_DONE syncpts have been sent to Host1x
VI_MWA_REQ_DONE	0x04	2	Predefined Operation Done for All Requests for a Frame condition. The VI2 generates this syncpt when all requests for a captured frame from the camera (PPA) to memory are completed.
VI_MWB_REQ_DONE	0x05	2	Predefined Operation Done for All Reqs for a Frame condition. The VI2 generates this syncpt when all requests for a captured frame from the camera (PPB) to memory are completed.
VI_MWA_ACK_DONE	0x06	2	Predefined Operation Done for all WrAcks for a Frame condition. The VI2 generates this syncpt when all WrAcks for a captured frame from the camera (PPA) to memory are received.
VI_MWB_ACK_DONE	0x07	2	Predefined Operation Done for all WrAcks for a Frame condition. The VI2 generates this syncpt when all WrAcks for a captured frame from the camera (PPB) to memory are received.
VI_ISPA_DONE	0x08	2	Predefined Operation Done condition. The VI2 generates this syncpt when it finishes sending a frame from the configured camera to ISPA
VI_CSI_PPA_FRAME_START	0x09	2	Valid SOF has been received by the camera (PPA) input. This SOF is detected at the VI2 input.
VI_CSI_PPB_FRAME_START	0x0A	2	Valid SOF has been received by the camera (PPB) input. This SOF is detected at the VI input.
VI_CSI_PPA_LINE_START	0x0B	2	Received an SOL indication on the camera (PPA) input stream. This SOL is detected at the VI2 input.
VI_CSI_PPB_LINE_START	0x0C	2	Received an SOL indication on the camera (PPB) input stream. This SOL is detected at the VI2 input.
VI_VGP0_RECD	0x0D	2	Rising edge detection on VGPx (VGPx being selected by a configuration register)
VI_VGP1_RECD	0x0E	2	Rising edge detection on VGPy (VGPy being selected by a configuration register)
VI_ISPB_DONE	0x0F	2	Predefined Operation Done condition.

Condition	Value	FIFO Depth	Description
			The VI2 generates this syncpt when it finishes sending a frame from the configured camera to ISPB
VI_PPA_TSC_MW_DONE	0x18	-	Reserved
VI_PPB_TSC_MW_DONE	0x19	-	Reserved
<reserved>	0x1A-0xFF	-	Reserved

28.1.4.3 Implementation Details of Configuration Registers

When the Host1x writes to any of the functional mode registers, the configuration value is immediately visible. So the software should write to such registers when the CSI/VI2 pipeline is idle. Software has visibility when the CSI/VI2 units are idle because it is not waiting for any syncpts.

When the VI2 unit receives the SOF from the CSI unit, it raises the syncpt corresponding to SOF_RECD on the appropriate syncpt index. This indicates to the software that the CSI and VI2 units are currently capturing a frame. At this time, software sends out the operational parameters for the next frame. These register writes are written to the shadow register. After the shadow register writes are complete, software indicates that the shadow registers update is done by writing to the VI_CSI_[0..1]_SINGLE_SHOT_CAPTURE register. After the single shot capture trigger is set, all shadow copies are now programmed and the VI2 will wait for the next EOF for transferring the active copy. When it receives the EOF and the previous frame is error free and correct, it updates the active registers from the shadow copy.

The VI2 marks all writes to memory as writes with ACK required. It also maintains a counter to keep track of number of writes outstanding. When the EOF is received from the CSI and there are no WrAcks pending, the VI2 generates the MWA_REQ_DONE or MWB_REQ_DONE syncpt to Host1x. Software uses this syncpt to pass this memory buffer to the next unit which will process the frame. In case the VI2 is configured to send the frame to ISP, it raises the syncpt ISPA_DONE when the EOF is sent. Depending on the configuration of HOST1X_UPDATE register configuration, these syncpts are either raised when the frame is known to be good or when any frame is sent to memory/ISP.

28.1.4.4 Error Status and Interrupts

This section discusses the reporting of various error conditions that can occur with the VI2 during the operation of the camera subsystem. The VI_CSI_[0..1]_ERROR_STATUS and VI_CSI_[0..1]_ERROR_INT_MASK functional registers are expected to be active for the camera use session. The remaining VI2 functional registers are described in the following table.

The VI_CSI_[0..1]_ERROR STATUS is a read-only register that indicates whether an error has occurred. All of the conditions defined in this register can be programmed to generate an interrupt using the VI_CSI_[0..1]_ERROR_INT_MASK as described in the following table.

Table 114: VI Error Status and Interrupt Mask Registers Functional Register Definitions

Register	Field	Bits	Reset Value	Description
VI_CSI_[0..1]_ERROR_STATUS	LINE_WIDTH_SHORT_ERROR:	0	X	Flagged if the frame line width is smaller than WIDTH. Write 1 to clear.
	LINE_WIDTH_LONG_ERROR:	1	X	Flagged if the frame line width is larger than WIDTH. Write 1 to clear
	FRAME_HEIGHT_SHORT_ERROR	2	X	Flagged if the frame height is smaller than HEIGHT. Write 1 to clear

Register	Field	Bits	Reset Value	Description
	FRAME_HEIGHT_LONG_ERROR	3	X	Flagged if the frame height is larger than HEIGHT. Write 1 to clear
	CSI_FRAME_ERROR	4	X	Flagged if the EOF field from the CSI has FRAME_ERROR bit set
	WATCHDOG_INT	5	X	This event occurs when Watchdog timer expires and the EOF is not received on the CSI2VI interface
VI_CSI_[0..1]_ERROR_INT_MASK	LINE_WIDTH_SHORT_INT_MASK	0	0x0	Line width short error interrupt mask
	LINE_WIDTH_LONG_INT_MASK	1	0x0	Line width long error interrupt mask
	FRAME_HEIGHT_SHORT_INT_MASK	2	0x0	Frame height short error interrupt mask
	FRAME_HEIGHT_LONG_INT_MASK	3	0x0	Frame height long error interrupt mask
	CSI_FRAME_ERROR_INT_MASK	4	0x0	CSI error interrupt mask
	WATCHDOG_INT_MASK:	5		Watchdog Timer 0 Interrupt Mask. This controls interrupts when a Watchdog trigger event occurs for the CSI stream. 0 = Disabled 1 = Enabled

28.1.4.5 Watchdog Counter

To detect catastrophic error conditions in which the CSI2 or VI2 has ceased operating, there are two Watchdog timers provided with the VI2 unit. The two WD timers are associated with the two image data streams being presented to the VI2 through the CSI PPA and PPB ports. Once the WD timer is configured for an image data stream, the timer must be prevented from expiring by the hardware. If the hardware does not disable the timer before the count expires, an interrupt is generated.

The WD timer counter is initialized to the value in the PERIOD register on the arrival of the SOF packet at the CSI to the VI interface for the active stream. The counter then proceeds to count down one count for every pixel clock. If the count reaches zero, with the WATCHDOG_INT_MASK bit set in the VI_CSI_[0..1]_ERROR_INT_MASK register, an interrupt is generated.

If an EOF packet is received at the CSI0 or CSI1 port, the countdown is stopped. In normal operation, this prevents the counter from reaching zero and generating an interrupt.

The ISP also provides a WD timer system.

28.1.5 Programming Sequences

The programming of the VI2 to initiate data capture involves setting up the image sensor, the CSI2 unit, the VI2 unit, and the ISP unit if the data is targeted for processing by the ISP. There are several different modes of image capture supported by the camera subsystem including progressive frame capture, interlaced frame capture, and interleaved frame capture. The capture modes may be set up as either single or dual stream to support the various user models, with the exception of the interleaved still capture. In the interleaved capture mode, both streams are assumed to be active.

28.1.5.1 Basic Progressive Frame Capture Sequence

The basic progressive frame capture programming sequence supports a number of user models involving both single and dual stream image captures. The basic sequence for the progressive capture is as follows:

1. Program the sensor or sensors in the dual stream use case.
2. Program the CSI and VI with parameters that do not change every frame.
3. Program Frame #1 specific parameters for the CSI and VI.
4. Enable CSI pixel parsers after completing PPA and/or PPB setup.
5. Arm the syncpt.
6. Trigger single shot by setting CAPTURE in the VI_CSI_[0..1]_SINGLE_SHOT register

   If (Outstanding capture requests == 1)

   Wait for SOF

   Else if (Outstanding capture requests > 1)

   Wait for MW_REQ_DONE
7. Program Frame #2 specific parameters for CSI and VI.
8. Repeat from step 5 for the next frames.

28.1.5.2 Basic Interlaced Frame Capture Sequence

The interlaced capture programming sequence is similar to the basic progressive capture sequence. However, since the interlaced capture results in the capture of a top and bottom field pair for each frame, there are some differences in how the hardware manages the syncpt generation and the single shot generation. Because the frame capture consists of a frame pair, both the syncpt and single shot are also managed based on frame pairs. The basic sequence for the interlaced capture is as follows:

1. Program the sensor or sensors in dual stream use case
2. Program the CSI and VI with parameters that do not change every frame.
3. Program Frame #1 Top/Bottom Field specific parameters for the CSI and VI.
4. Program the CSI TOP_FIELD_FRAME and TOP_FIELD_FRAME_MASK
5. Enable CSI Pixel Parser after completing the PPA and/or PPB setup
6. Arm syncpts for the field pair capture
7. Trigger single shot by setting CAPTURE in the VI_CSI_[0..1]_SINGLE_SHOT register

   If (Outstanding capture requests == 1)

   Wait for PP[A..B]_ FRAME_START

   Else if (Outstanding capture requests > 1)

   Wait for MW_REQ_DONE
8. Program Frame #2 specific parameters for CSI and VI.
9. Repeat from step 6 for next frames.

Step 4 is unique to the interlaced capture and is necessary for generating the top/bottom field signaling for the VI2. This signaling manages the data flow to memory and handling of field drop error conditions. The SOF syncpt is generated only when a top field is signaled on the CSI to VI interface, and the MW_REQ_DONE syncpt is generated only after a top and bottom field are paired and written to memory. The VI2 generates a single shot trigger for the top and bottom field when the

interlaced mode is enabled. In addition, the VI2 provides for a frame-dropping mechanism which automatically re-issues a single shot to capture another field if a frame is dropped.

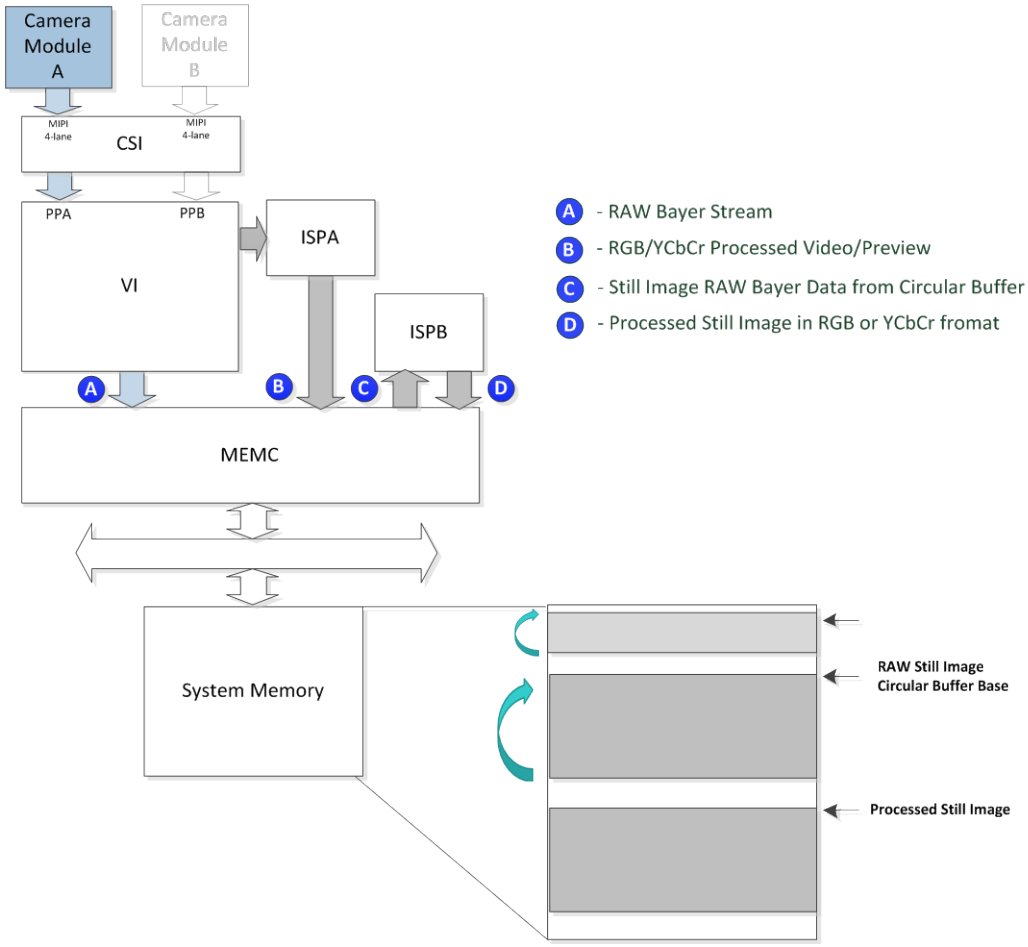
28.1.6 Frame Capture Modes

There are several frame capture scenarios available within the camera subsystem, which utilize various resources within the VI2. The modes include scene preview or simple progressive video capture, multi camera progressive or interlaced video capture, high-resolution still capture, high-resolution zero shutter lag still capture, and an 8-lane interleaved capture.

28.1.7 High Resolution and ZSL Still Image Capture

The high resolution and ZSL still capture models utilize the VI2 resources for providing a path to system memory for Raw Bayer image data while simultaneously maintaining an image data stream to the ISP(s) for preview. The following figure shows the active data paths during mono ZSL capture mode.

Figure 96: Zero Shutter Lag Mode

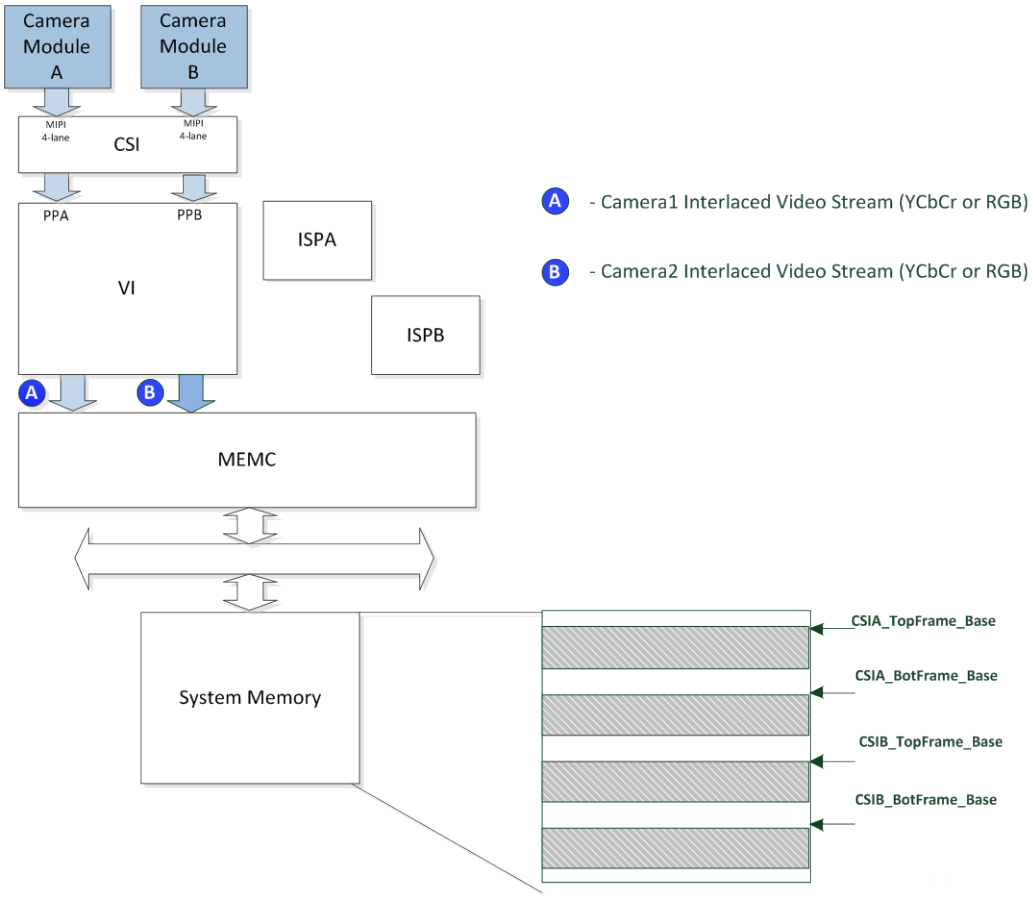


28.1.8 Interlaced Video Capture

The interlaced capture model provides for the management of interlaced video from an external camera module. The interleaved image data is presented to the CSI in a sequence of odd and even frames which represent the top and bottom fields in the interleaved composite frame. Since the interleaved image frames are not targeting raw Bayer processing by the ISP, the data formats are limited to either be RGB or YCbCr. The VI2 capture of interlaced video allows up to two camera

pixel streams and associated channel pointers frame pointers to be managed simultaneously. The interlaced video capture mode is shown in the following figure.

Figure 97: Interlaced Capture Mode



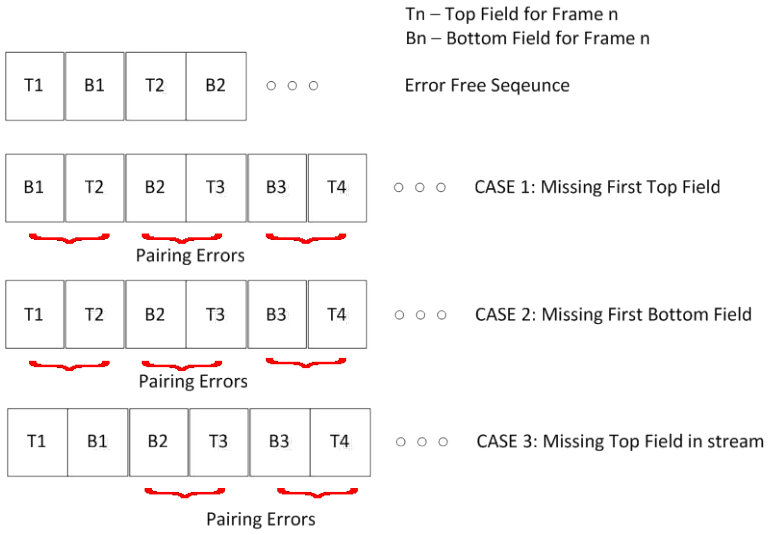
Twelve (12) pointers are available for managing the interleaved image streams. Three are available for the top field and three are available for the bottom fields for each of the two streams. All three pointers are used for each stream when the data is YCbCr422 and the target is planar, otherwise either two or only one are needed for semi-planar and non-planar formats, respectively.

The CSI unit signals the VI2 when a frame is either the top or bottom field by setting or clearing the LSB on the CSI[A,B]2VI_DATA bus when the SOF is being sent. The VI2 uses the signaling to provide for switching between the top and bottom field source and stride registers setting used for the address calculations. The signals are managed through the CSI by programming the CSI_PPA_TOP_FIELD_FRAME_MASK and CSI_PPA_TOP_FIELD_FRAME parameters. The actual frame number used in the associated calculation is taken from the WC field of the Frame Start short packet.

28.1.8.1 Interlaced Capture Error Handling

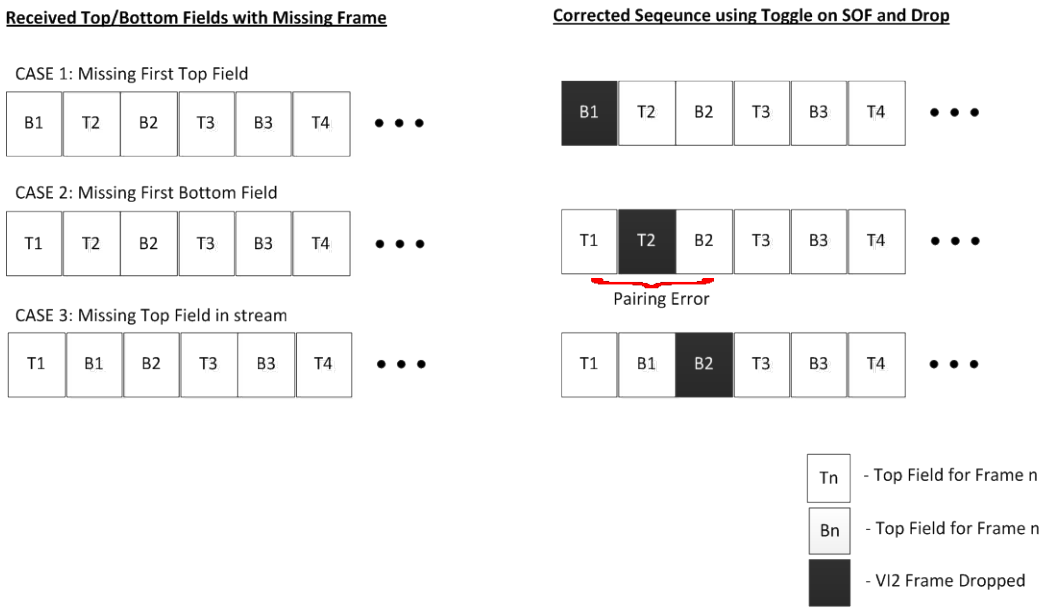
The interlaced capture generates two frames of data, one for the top field and one for the bottom field. It is important that the pairing of the frames in the top and bottom field buffers is accurate since any mismatch could result in artifacts that could propagate as errors in future frames captured in the video sequence. The issue is manifested with the occurrence of dropped frames either at the camera module or in the CSI2. If a frame is lost, several pairing errors are possible.

Figure 98: Interleaved Field Pairing Errors due to Dropped Frames



In order to manage the dropped frame event, a flag is maintained in hardware that toggles on every SOF received from the CSI while in interlaced video mode. This indicates whether the frame is even or odd. The VI2 also compares the state of the TOP field bit during the SOF. If the flag is odd and the top field bit is not set, the received field is then dropped. In a similar fashion, a missing bottom field results in a top field being dropped. The toggle and drop correction are illustrated in the following figure.

Figure 99: Error Correction for Interlaced Capture



The loss of a bottom field results in a single pairing error but is not propagated into the following field pair. It is considered to be an acceptable error tradeoff.

28.1.8.2 Interlaced Single Shot and Sync Points

The programming sequence for the interlaced capture is basically the same as for the progressive capture. The primary difference between the two modes is that the interlaced mode operates on the concept of top and field pairs. The VI2 manages the frame single shot trigger and EOF events based on the successful pairing of top and bottom fields.

The VI_CSI_[0..1]_SINGLE_SHOT is programmed once for each top and bottom field pair in the interlaced mode of frame capture and hardware manages the generation of a single shot trigger for both the top and bottom field capture. In the event that a top field is not signaled by the CSI following the issue of the single shot capture, the frame is dropped and the single shot is re-issued until a top field is indicated. Once a top field has been successfully captured and an EOF received, the VI2 then issues a single shot capture for the bottom field. In a similar fashion with a missing top field first, the single shot is re-issued if a bottom filed is not signaled after the top field was successfully captured.

Sync points are also managed in a field pair fashion making it necessary to arm a sync point once for the frame event. For example, the VI_CSI_PP[A..B]_FRAME_START sync points are only returned on a successful start of a top field capture and similarly the VI_MW[A..B]_REQ_DONE sync points are only returned after the successful completion of a bottom field capture in a top/bottom field pair.

28.1.9 Stereo Still and Video Capture

The stereo still and video capture modes can utilize ISPA and ISPB to process the left and right camera streams simultaneously in support of preview, video capture, and the combination of preview and high-resolution still processing. The stereo capture models should be managed carefully to achieve maximum throughput and minimize system loading. The stereo capture modes are listed in the following table.

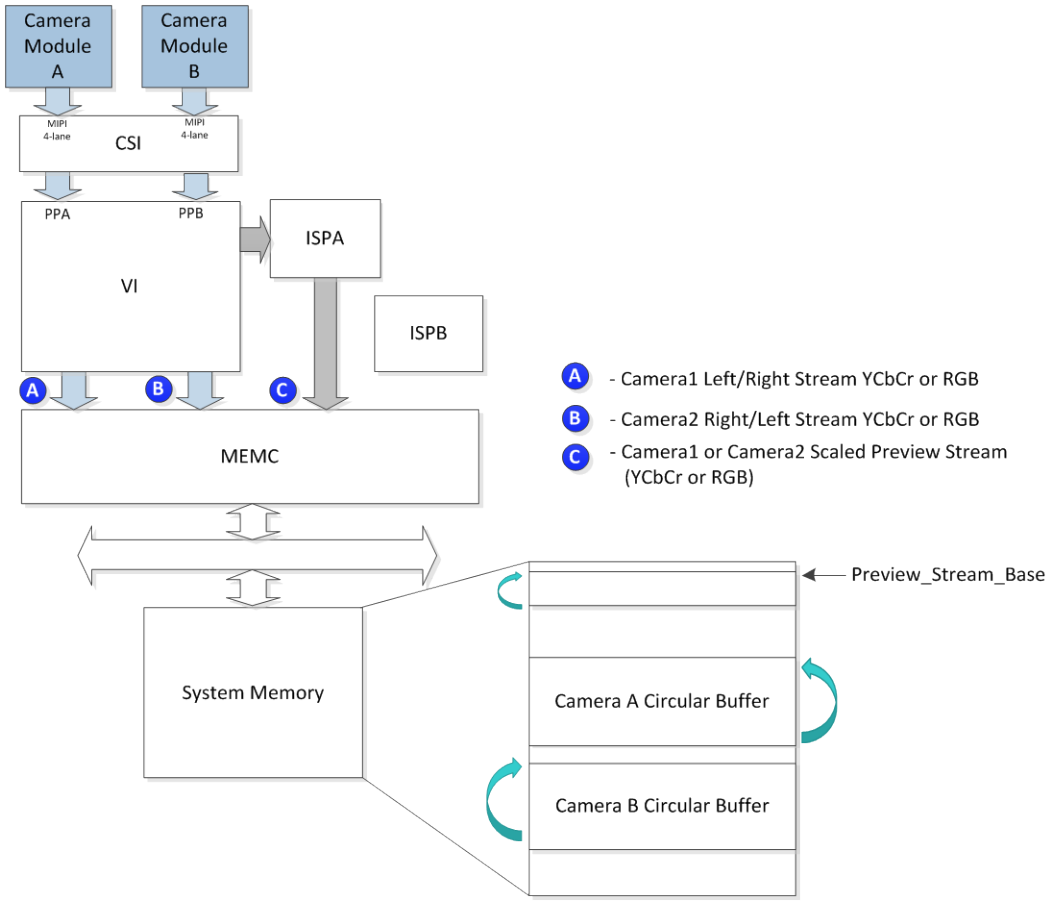
Table 115: Stereo Image/Video Capture Modes

Stereo Mode	Data Type	
Stereo Preview	YUV/RGB/RAW	Both Streams Active; ISPA or ISPB may be used for preview management; Should provide L/R preview switching; Raw data can be processed through the ISP or YUV. RGB data may be enhanced or delivered to memory through the MCCIF.
Stereo Preview for ZSL Still	YUV/RGB	Both Streams Active; ISPA or ISPB may be used for preview management; Should provide L/R preview switching; High resolution stills are delivered to the circular buffer in system memory through the MCCIF
	RAW	Both Streams Active; ISPA or ISPB used for Raw processing of one stream. Should provide L/R preview switching. High resolution Raw stills are delivered to the circular buffer in system memory through the MCCIF. During capture, the idle ISP may be used for processing so that preview can be continuous using the other ISP.
Stereo Preview + Video Capture	YUV/RGB/RAW	Both Streams Active; ISPA or ISPB may be used for preview management. Should provide L/R preview switching. Raw data may be processed through the ISP or YUV. RGB data may be enhanced or delivered to memory through the MCCIF.

There are three basic modes of operation when engaging the stereo capture user model: the preview of the image data for scene composition, the capture of a high resolution stereo pairs, and the capture of stereo video.

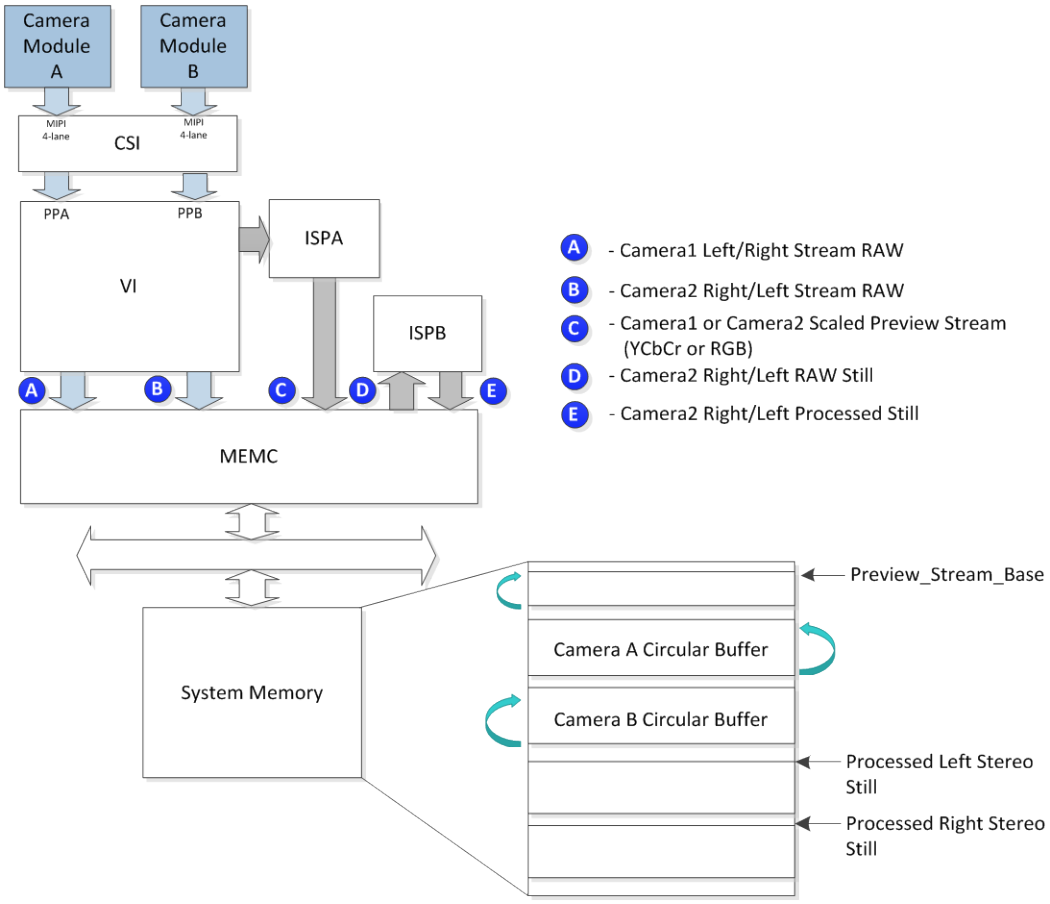
The still and video capture models for YUV or RGB processed data from the camera module may utilize the ISPA or ISPB resource as a scaling and image enhancement engine by providing the image data from either the left or right camera directly to the ISP. In most cases, the capture of processed data in a YUV or RGB format will be written directed to memory where it can be accessed by various resources in the system. The requirement for different resolutions for LCD display and the frame capture makes this option attractive since it can help to minimize memory traffic. A ZSL stereo capture sequence where both high-resolution streams are being stored in separate circular buffers for the left and right channels is illustrated in the following figure.

Figure 100: Zero Shutter Stereo Capture for YUV or RGB Image Data



The still and video capture models for Raw data utilize both ISPA and ISPB simultaneously under most conditions. During the stereo video capture, the image streams being captured by the left and right camera modules can each be assigned to either ISPA or ISPB, allowing a preview to be extracted from either processed stream with the stereo video being available to the compression resources. The ZSL stereo capture model also utilizes both ISPA and ISPB, however, only one ISP is used for managing the preview, and the second is idle until a still capture is initiated. Both high-resolution Raw streams are provided to system memory through the MCCIF to maintain the circular buffer for the left and right channels. Once the ZSL still capture is initiated, the idle ISP can be utilized to process the still images while the other is available to maintain a preview and allow a new ZSL sequence to be initiated with little delay. The basic sequence for the Raw stereo capture is shown in the following figure.

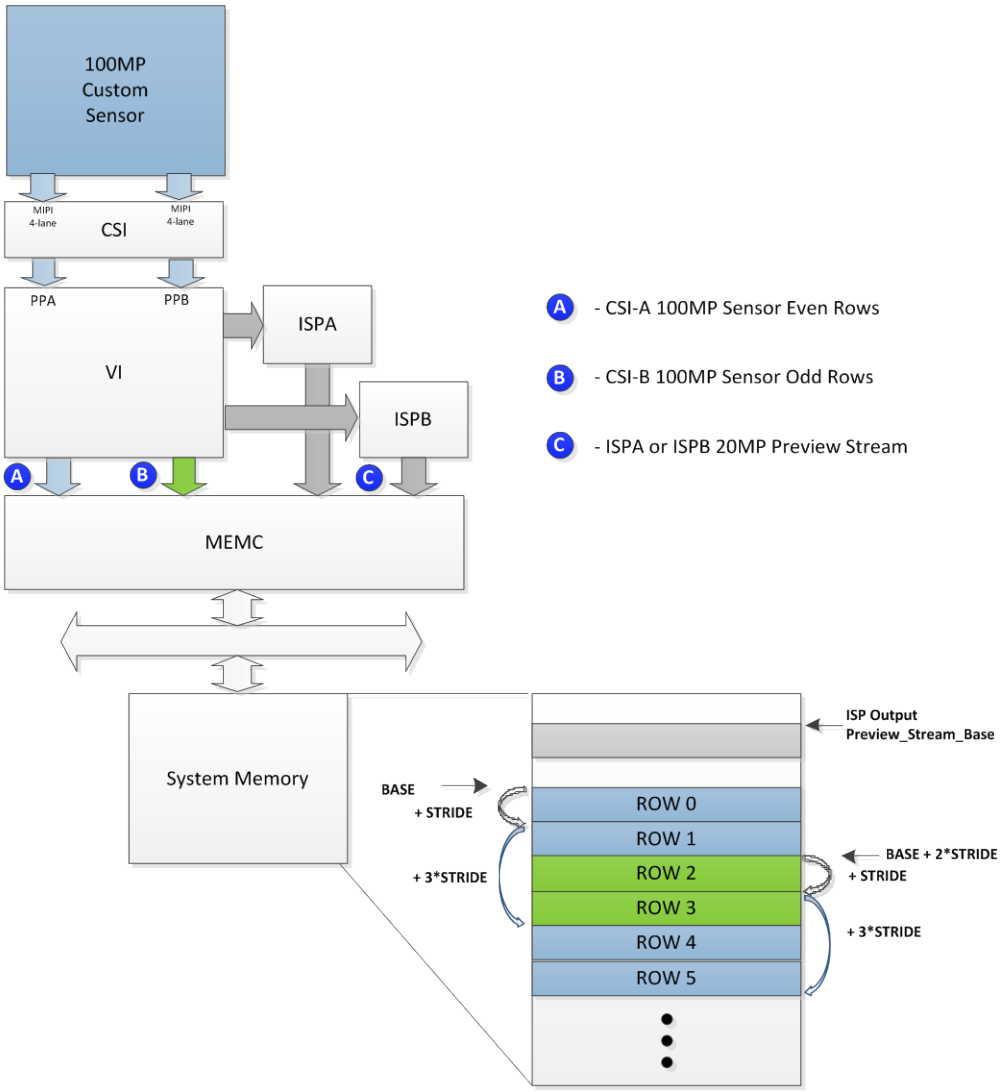
Figure 101: Zero Shutter Stereo Capture for Raw Image Data



28.1.10 Interleaved Capture

The interleaved capture mode supports management of pixel data delivered using all 8 available MIPI lanes. In this mode of operation, image data is delivered across all 8 lanes with two rows being provided simultaneously to the CSI-A and CSI-B MIPI ports. In this mode, the VI2 manages the interleaving of the even and odd rows when writing the data to system memory. The sensor interfaces to the CSI unit through the MIPI ports CSI-A and CSI-B using all four lanes for each interface. The following figure illustrates the interleaved capture model.

Figure 102: Interleaved Capture Model

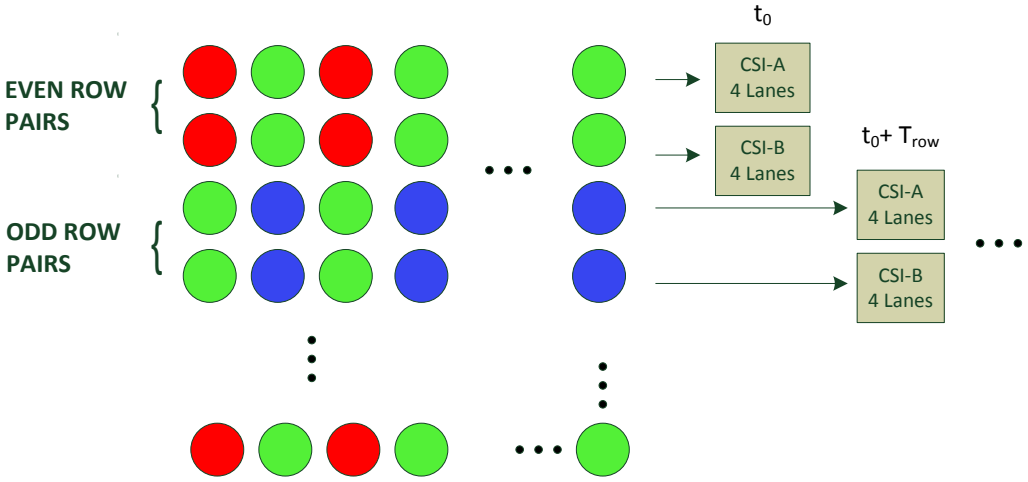


28.1.10.1 Interleaved Capture Model

The interleaved capture model supports image data that is presented as a standard RGGB Bayer pattern with either the even or odd rows available through CSI-A and the odd or even rows through CSI-B. The VI2 provides the RAW10 image data to system memory.

Utilizing both of the CSI 4-lane interfaces allows an effective still capture data rate to be doubled since all 8 MIPI lanes are active simultaneously. In this mode, the data is presented to the CSI interface in a unique fashion with even row pairs, both consisting of Red/Green pixels presented to CSI-A and CSI-B followed by the odd row pairs consisting of Green/Blue pixels.To provide a standard RGGB Bayer pattern in memory, an interleaving mechanism is provided as the data is written to memory. The following figure illustrates the even and odd row pair's presentation to the CSI.

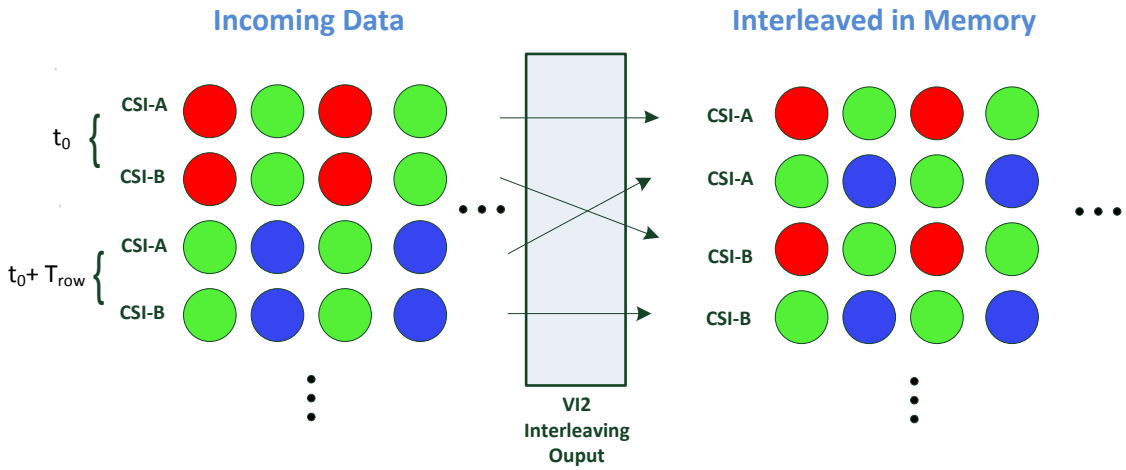
Figure 103: Interleaved Capture Data Presentation



28.1.10.2 Dual-Channel Interleaving

The interleaved still image is delivered in even and odd row pairs resulting in two Red/Green or two Blue/Green rows being presented to memory at the same time. The preparation of the data for processing involves interleaving the rows so that a Bayer pattern organization of the row data is available in system memory. To accomplish this, the VI2 provides a special interleaved mode in which the line counters for the two streams are incremented in a predetermined sequence. The following figure illustrates the interleaving of the incoming data stream in system memory.

Figure 104: Dual-Channel Interleaving During Write to Memory

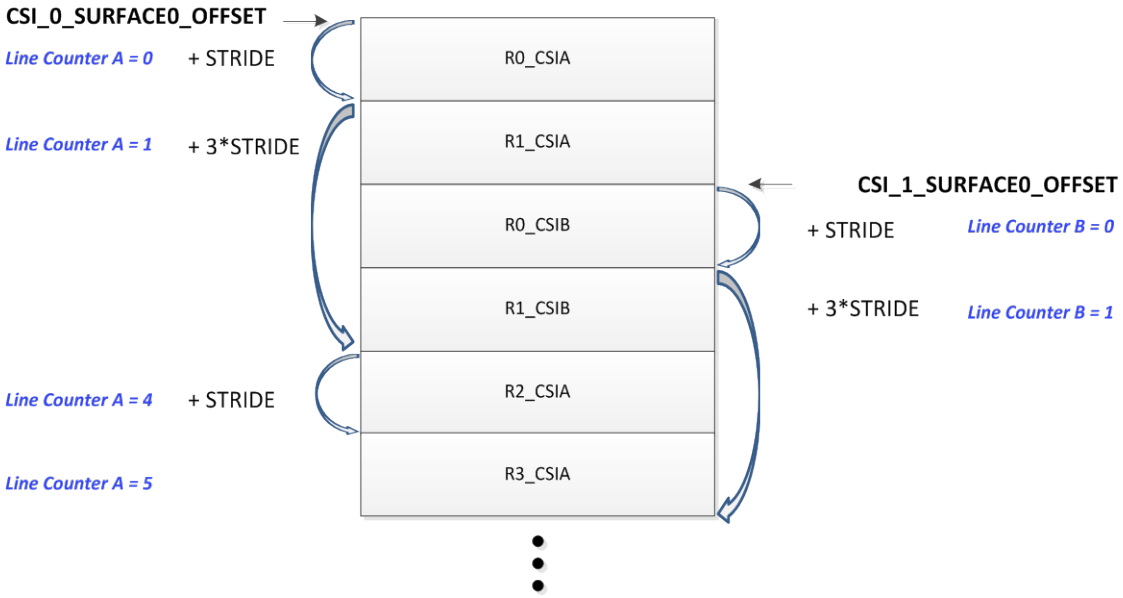


The interleaved mode is enabled by setting the INTERLEAVING_MODE bit in the VI_CSI_[0..1]_IMAGE_DEF registers. The mode setting changes the normal operation of the line counters used to calculate the memory address after a line is written to increment in a line pair skipping fashion as opposed to the normal mode of single increments.

Captured Row delivered by CSIA or CSIB	NORMAL MODE LINE COUNTER SEQUENCE	INTERLEAVED MODE LINE COUNTER SEQUENCE
Row 0	0	0
Row 1	1	1
Row 2	2	4
Row 3	3	5
Row 4	4	8
Row 5	5	9
.	.	.
.	.	.
.	.	.
Row (Height-1)	HEIGHT-1	HEIGHT-1

The base address and stride for each of the channels are programmed prior to initiating the image capture. Once the image capture is initiated, the line counters for the two channels will sequence in a row pair fashion as illustrated in the following figure.

Figure 105: Dual Channel Interleaving in Memory for T_R16_I



The RAW10 format is used for the interleaving capture and may be stored in either the T_R16_I or the T_L10_L10_L10_X2 formats. In addition to the INTERLEAVING_MODE selection, six registers must be programmed to manage the writing of the data to memory. Since the VI2 provide for memory writes on a 16 byte boundary, it is necessary to program the above registers accordingly

Table 116: Dual-Channel Interleaving Mode Address and Stride Programming

Register	Data Format	
VI_CSI_0_SURFACE0_OFFSET_LSB VI_CSI_0_SURFACE0_OFFSET_MSB	T_L16_I or T_L10_L10_L10_X2	Base Address for the first even row pair still image in system memory
VI_CSI_1_SURFACE0_OFFSET_LSB VI_CSI_1_SURFACE0_OFFSET_MSB	T_L16_I or	Base Address for the odd even row pair still image in system memory

Register	Data Format	
	T_L10_L10_L10_X2	VI_CSI_0_SURFACE0_OFFSET_LSB + 2 * STRIDE.
VI_CSI_0_SURFACE0_STRIDE0 VI_CSI_1_SURFACE0_STRIDE0	T_L16_I	STRIDE = 2 * IMAGE_WIDTH
	T_L10_L10_L10_X2	STRIDE = (4/3) * IMAGE_WIDTH

28.1.11 TrustZone Support

The VI2 allows management of several CSI registers associated with memory transactions and resets as either freely programmable in the system or only programmable by secure world functions. The registers supporting the secure programming feature include the reset and base, stride, and image definition registers for CSI0 and CSI1. These Trustzone support registers are listed below.

- VI_CSI_[0,1]_ SW_RESET
- VI_CSI_[0,1]_ IMAGE_DEF
- VI_CSI_[0,1]_SURFACE0_OFFSET_MSB
- VI_CSI_[0,1]_SURFACE0_OFFSET_LSB
- VI_CSI_[0,1]_SURFACE1_OFFSET_MSB
- VI_CSI_[0,1]_SURFACE1_OFFSET_LSB
- VI_CSI_[0,1]_SURFACE2_OFFSET_MSB
- VI_CSI_[0,1]_SURFACE2_OFFSET_LSB
- VI_CSI_[0,1]_SURFACE0_BF_OFFSET_MSB
- VI_CSI_[0,1]_SURFACE0_BF_OFFSET_LSB
- VI_CSI_[0,1]_SURFACE1_BF_OFFSET_MSB
- VI_CSI_[0,1]_SURFACE1_BF_OFFSET_LSB
- VI_CSI_[0,1]_SURFACE2_BF_OFFSET_MSB
- VI_CSI_[0,1]_SURFACE3_BF_OFFSET_LSB
- VI_CSI_[0,1]_SURFACE0_STRIDE
- VI_CSI_[0,1]_SURFACE1_STRIDE
- VI_CSI_[0,1]_SURFACE2_STRIDE
- VI_CSI_[0,1]_SURFACE_HEIGHT0

Programming control of the secure CSI registers is based on the programmed output security PMC's APBDEV_PMC_STICKY_BITS_0 field: CSI_SECURE_ENABLE and the Trustzone secure mode being active or inactive. Once the CPU is context switched from the normal operation “non-secure world” to the “secure world” (indicated to the rest of the system by the TrustZone _NS (non-secure bit) being deasserted, for example, there is a host1x trustzone_nonsecure signal), the VI allows for writes to the CSI registers when the sticky bit register is asserted. If the sticky bit is asserted, but the _NS signal is asserted (that is, the CPU is operating in the non-secure world), then the VI’s CSI registers are not writable. If the sticky bit is not asserted at all, then the VI’s CSI registers can be written to normally.

28.2 Data Formatting

The following sections describe the CSI Pixel formats, ISP pixel formatting, and the various formats and packing options for delivering data to system memory.

28.2.1 CSI Input Data Formats

The CSI provides data to the VI2 in one of several possible formats. The following table describes the different formats presented to the VI2 through the CSI interfaces.

Table 117: CSI to VI2 Pixel Formats

Format Name		23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
6-bit raw or		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
7-bit raw or		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D6	D5	D4	D3	D2	D1	D0
8-bit raw or		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D7	D6	D5	D4	D3	D2	D1	D0
10-bit raw or		0	0	0	0	0	0	0	0	0	0	0	0	0	0	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
12-bit raw		0	0	0	0	0	0	0	0	0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
14-bit raw		0	0	0	0	0	0	0	0	0	0	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
8-bit arbitrary/embedded		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D7	D6	D5	D4	D3	D2	D1	D0
16-bit RGB (RGB565)		B4	B3	B2	B1	B0	0	0	0	G5	G4	G3	G2	G1	G0	0	0	R4	R3	R2	R1	R0	0	0	0
15-bit RGB (RGB555)		B4	B3	B2	B1	B0	0	0	0	G4	G3	G2	G1	G0	0	0	0	R4	R3	R2	R1	R0	0	0	0
24-bit RGB (RGB888)		B7	B6	B5	B4	B3	B2	B1	B0	G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0
12-bit RGB (RGB444)		B3	B2	B1	B0	0	0	0	0	G3	G2	G1	G0	0	0	0	0	R3	R2	R1	R0	0	0	0	0
18-bit RGB (RGB666)		B5	B4	B3	B2	B1	B0	0	0	G5	G4	G3	G2	G1	G0	0	0	R5	R4	R3	R2	R1	R0	0	0
YUV422 8 -bit	1st CLK	V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0	Y07	Y06	Y05	Y04	Y03	Y02	Y01	Y00
	2nd CLK	V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
YUV422 10-bit	1st CLK	V9	V8	V7	V6	V5	V4	V3	V2	U9	U8	U7	U6	U5	U4	U3	U2	Y09	Y08	Y07	Y06	Y05	Y04	Y03	Y02
	2nd CLK	V9	V8	V7	V6	V5	V4	V3	V2	U9	U8	U7	U6	U5	U4	U3	U2	Y19	Y18	Y17	Y16	Y15	Y14	Y13	Y12
YUV420 8-bit (legacy)	Odd Line 1st CLK	0	0	0	0	0	0	0	0	U7	U6	U5	U4	U3	U2	U1	U0	Y07	Y06	Y05	Y04	Y03	Y02	Y01	Y00
	Odd Line 2nd CLK	0	0	0	0	0	0	0	0	U7	U6	U5	U4	U3	U2	U1	U0	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
	Even Line 1st CLK	V7	V6	V5	V4	V3	V2	V1	V0	0	0	0	0	0	0	0	0	Y07	Y06	Y05	Y04	Y03	Y02	Y01	Y00
	Even Line 2nd CLK	V7	V6	V5	V4	V3	V2	V1	V0	0	0	0	0	0	0	0	0	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
YUV420 8-bit (CSPS)	Odd Line	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
	Even Line 1st CLK	V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0	Y07	Y06	Y05	Y04	Y03	Y02	Y01	Y00
	Even Line 2nd CLK	V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
YUV420 10-bit (CSPS)	Odd Line	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Y9	Y8	Y7	Y6	Y5	Y4	Y3	Y2
	Even Line 1st CLK	V9	V8	V7	V6	V5	V4	V3	V2	U9	U8	U7	U6	U5	U4	U3	U2	Y9	Y8	Y7	Y6	Y5	Y4	Y3	Y2
	Even Line 2nd CLK	V9	V8	V7	V6	V5	V4	V3	V2	U9	U8	U7	U6	U5	U4	U3	U2	Y9	Y8	Y7	Y6	Y5	Y4	Y3	Y2

The formats include various flavors of RAW, RGB, and YCbCr in addition to 8-bit arbitrary or embedded data. The bits marked ‘0’ are actually don’t-care for VI2 and are ignored by the VI2 unit and the odd line in the above table refers to the first line from the CSI unit.

28.2.2 Raw Pixel Data Formatting

The Raw pixel data may be delivered to the ISP, system memory, or both the ISP and system memory simultaneously. In order to interface to the ISP efficiently, pixels are prepared for the ISP pipeline by converting to the unsigned offset binary representation of the form:

I.F

Where:

I is the number of integer bits.

F is the number of fractional bits.

The zero offset shall have a value of $2^{(F + I - 2)}$. For the ISP, it is suggested that the values of I and F should be:

I = 1, F = 13, offset = 4096_{10}

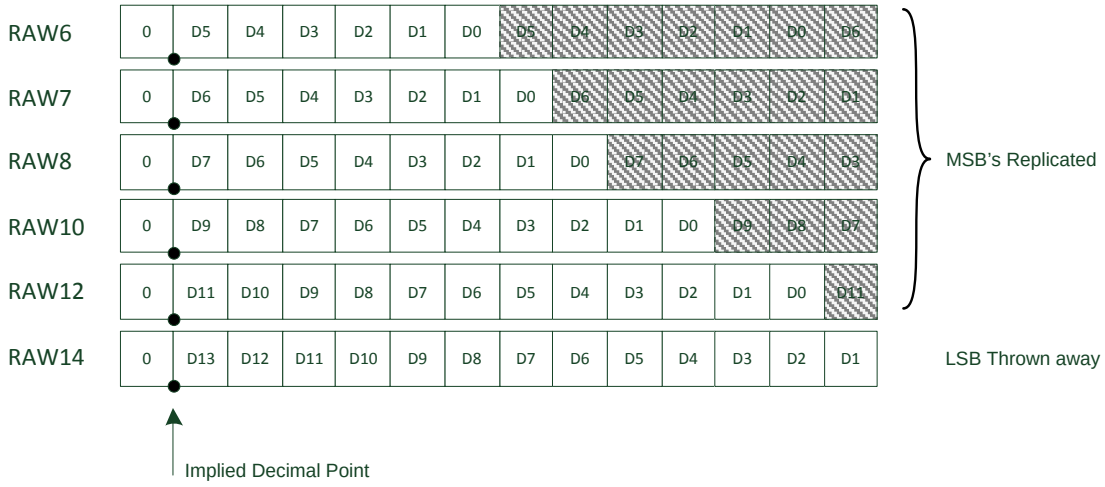
The formatting of the captured RAW data to the 14-bit I.F format requires several steps to be performed. Data may also be stored to memory either simultaneous with the ISP presentation or alone. When storing data to memory, either the formatted data or the original data may be stored to memory. Since the VI2 RAW data streams are all targeted for processing by the ISP, the storing of data which has formatting provided within the VI2 pipeline is the default behavior. Optionally the formatting may be bypassed by setting the BYPASS_PXL_TRANSFORM in the VI_CSI_[0..1]_IMAGE_DEF register.

The formatting steps for preparation of RAW pixel data for the ISP pipeline and/or memory are described in the following sections.

28.2.2.1 Raw Pixel Formatting to VI Pipeline

The Raw data received from the CSI unit is formatted as a 14-bit unsigned operand, U1.13. The data is scaled to provide align the MSB with the implied decimal point. When converting from a source data size that is smaller than the destination data size, the MSBs of the source are replicated to fill the missing LSBs in the destination. The bit replication is shown in the following figure.

Figure 106: VI2 Pipeline Formatting for U1.13 Raw Pixel Data

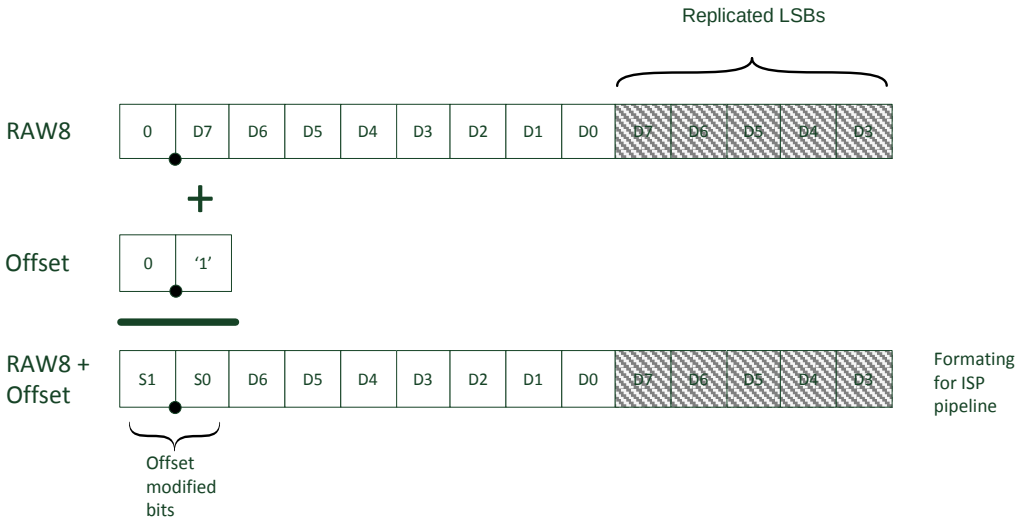


Since the RAW14 data would be wider than the ISP pipe when converting to the unsigned U1.13 format, the LSB is truncated.

28.2.2.2 RAW Pixel Formatting to ISP Pipeline

If the destination format is the pixel bus format, the data must be aligned with the implied decimal point position and the offset must be added. For example the formatting of RAW8 pixel data for the ISP pipeline is illustrated in the following figure.

Figure 107: RAW8 to ISP Pipeline Formatting Example



28.2.2.3 Raw Pixel Formatting to Memory

The raw pixel data may be presented to system memory in three possible formats specific to the input bit precision. The TL_8 formatting operation provides source to destination bit precision increase and decrease. All of the TL_8 formats are packed four per 32-bit word when stored to memory. The T_R16_I employs MSB replication based off of the RAW data previously formatted for the VI pipeline and a new format is introduced for packing three 10-bit raw pixels into a 32-bit word. The different options are summarized in the following table.

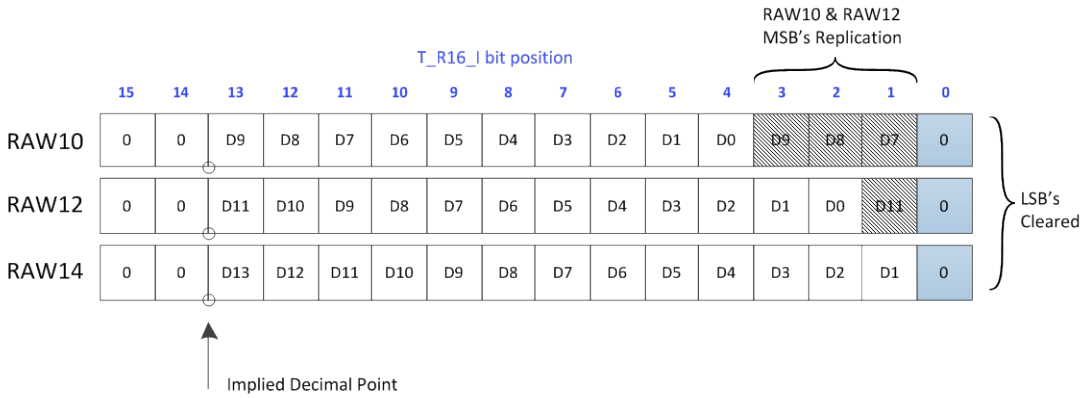
Table 118: Raw Supported Memory Formats

CSI Format Name	Pixel Format in Memory	bpp	Notes
RAW6	T_L8	8	
RAW7	T_L8	8	
RAW8	T_L8	8	
RAW10	T_L8	8	Drop 2 LSB
	T_R16_I	16	
	T_L10__L10__L10_X2	32	3 RAW10 Pixels Packed
RAW12	T_L8	8	Drop 4 LSB
	T_R16_I	16	
RAW14	T_L8	8	Drop 6 LSB
	T_R16_I	16	

T_R16_I Formatting for RAW10, RAW12, and RAW14

The formatting of the raw data from the VI2 pipeline for bit precisions above eight involves an additional MSB bit replicationI as illustrated in the following figure.

Figure 108: RAW10, RAW12, and RAW14 Formatting to Memory



T_L8_F Formatting for all Raw Pixel Precisions

The formatting for the 8-bit data is a straightforward mapping with either truncation or MSB replication being employed. The formatting for 8-bit data is illustrated in the following figure.

Figure 109: Raw Formatting to T_L8 Memory Format

T_L10__L10__L10_X2 Formatting for RAW10

The T_L10_L10_L10_X2 packed format is introduced to help reduce the burden on the system memory resources during high BW camera user models. In this format, three RAW10 pixels are packed per 32-bit word allowing an improvement in utilization to be realized. The memory efficiency is 93.75% compared to the T_R16_I format with 62.5%.

28.2.3 RGB Pixel Data Formatting

The VI2 receives RGB data from the CSI in one of several possible formats and provides for formatting and presentation to system memory. The RGB formats are represented through variation in the number of bits used to represent the red, green, and blue channels. The RGB memory formats and options for source to destination formatting are described in the following table.

Table 119: RGB Supported Memory Formats

CSI Format Name	Pixel Formats in Memory	bpp	Notes
RGB565	T_L8	8	(C1*R+C2*G+C3*B)/16
	T_A1B5G5R5; T_A1R5G5B5; T_B5G5R5A1; T_R5G5B5A1	16	
	T_B5G6R5; T_R5G6B5	16	
RGB555	T_L8	8	(C1*R+C2*G+C3*B)/16
	T_A1B5G5R5; T_A1R5G5B5; T_B5G5R5A1; T_R5G5B5A1	16	
	T_B5G6R5; T_R5G6B5	16	
RGB888	T_L8	8	(C1*R+C2*G+C3*B)/16
	T_A1B5G5R5; T_A1R5G5B5; T_B5G5R5A1; T_R5G5B5A1	16	
	T_A4B4G4R4; T_A4R4G4B4; T_B4G4R4A4; T_R4G4B4A4	16	
	T_B5G6R5; T_R5G6B5	16	
	T_A8B8G8R8; T_A8R8G8B8; T_B8G8R8A8; T_R8G8B8A8	32	
	T_A2B10G10R10; T_A2R10G10B10; T_B10G10R10A2; T_R10G10B10A2	32	
RGB444	T_L8	8	(C1*R+C2*G+C3*B)/16
	T_A1B5G5R5; T_A1R5G5B5; T_B5G5R5A1; T_R5G5B5A1;	16	
	T_A4B4G4R4; T_A4R4G4B4; T_B4G4R4A4; T_R4G4B4A4	16	
	T_B5G6R5; T_R5G6B5	16	
RGB666	T_L8	8	(C1*R+C2*G+C3*B)/16
	T_A1B5G5R5; T_A1R5G5B5; T_B5G5R5A1; T_R5G5B5A1	16	
	T_B5G6R5; T_R5G6B5	16	
	T_A8B8G8R8; T_A8R8G8B8; T_B8G8R8A8; T_R8G8B8A8	32	

Each of the capture RGB formats can be formatted by adjusting the precision of the channels, ordering the channels differently, or extracting a luminance approximation into a packed T_L8 format.

28.2.3.1 RGB Pixel Data Size Conversions

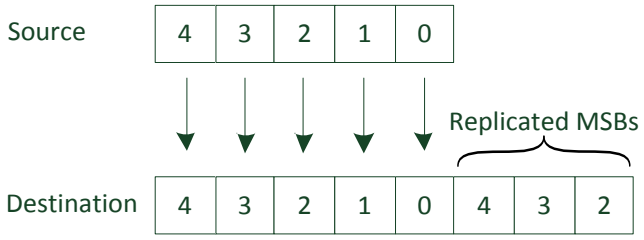
The VI2 supports the conversion of the received RGB data into other formats by either increasing or decreasing the precision of the data representing each of the channels. The following table describes the possible conversions.

Table 120: RGB Data Precision Adjustments Options

RGB Format Received	RGB Format Conversion (Precision Increase)	RGB Format Conversion (Precision Decrease)
RGB444	RGB555, RGB565	None
RGB555	RGB565	None
RGB565	None	RGB555
RGB666	RGB888	RGB555, RGB565
RGB888	RGB101010	RGB444,RGB555, RGB565, RGB666

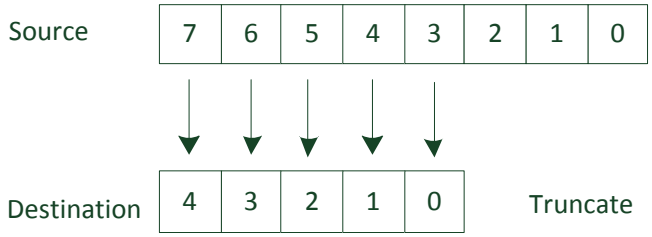
The RGB pixel data is converted to higher bit precision through replication of the MSBs in the source to fill in the missing LSBs in the destination in a similar fashion as the Raw data formatting for the ISP pipe. For example:

Figure 110: 5-bit to 8-bit Precision Increase



When converting from a source data size that is larger than the destination size, the destination is a truncated version of the source data. For example:

Figure 111: 8-bit to 5-bit Precision Decrease



28.2.3.2 RGB to Luma Data Formatting

In addition to providing format conversions, the RGB data may also be stored as luma data by approximating a luminance calculation. This is useful for applications that would otherwise have to compute the luminance channel from the RGB data in a separate step. The luma calculation is performed by multiplying each of the red, green, and blue channel components by a U0.6 coefficient as shown in the following pseudo code.

```
#define SHIFT      5
#define COEFF_BITS 6
```

```
 Y = ( (PIXEL.RED_CR * MW_R2Y_COEFF)
    + (PIXEL.GRN_Y  * MW_G2Y_COEFF)
   + (PIXEL.BLU_CB * MW_B2Y_COEFF) ) >> (SHIFT + COEFF_BITS);

if (Y < 0)   Y =   0;
```

```
    if (Y > 255) Y = 255;
```

28.2.3.3 RGB Pixel Formatting to ISP Pipeline

If the destination format is the pixel bus format, the data must be aligned with the implied decimal point position and the offset must be added. For example, the formatting of RGB565 pixel data for the ISP pipeline will involve formatting each of the color components into the 14-bit I.F format.

28.2.4 YUV Pixel Data Formatting

The VI2 provides for formatting and packing of YCbCr data received from the CSI and is responsible for its presentation to system memory or to the ISP for scaling or image enhancement. There are two formats captured: YUV422 and YUV420. The formatting provided for YUV capture data is shown in the following table.

Table 121: YUV Supported Memory Formats

CSI Format Name	Pixel Formats in Memory	bpp (per plane)	Notes
YUV422 8-bit	T_L8	8	Y component only
	T_Y8_U8__Y8_V8 [a] ; T_Y8_V8__Y8_U8 [a] ; T_U8_Y8__V8_Y8 [a] ; T_V8_Y8__U8_Y8 [a]	32	
	T_Y8___U8V8_N422	8	Semi-planar; Y+UV
	T_Y8___V8U8_N422	8	Semi-planar; Y+VU
	T_Y8___U8___V8_N422 [b]	8	Planar
YUV422 10-bit	T_L8	8	Y component only
	T_Y8_U8__Y8_V8 [a] ; T_Y8_V8__Y8_U8 [a] ; T_U8_Y8__V8_Y8 [a] ; T_V8_Y8__U8_Y8 [a]	32	
	T_Y8___U8V8_N422 [b]	8	Semi-planar; Y+UV
	T_Y8___V8U8_N422 [b]	8	Semi-planar; Y+VU
	T_Y8___U8___V8_N422 [b]	8	Planar
YUV420 8-bit legacy	T_L8	8	Y component only
YUV420 8-bit / 8-bit (CSPS)	T_L8	8	Y Component only.
YUV420 10-bit / 10-bit (CSPS)	T_L8	8	Y Component only.

The 8-bit YUV422 format is designated a primary format in the MIPI CSI specification and all others are considered as secondary formats. The supported pixel layout will be Pitch Linear or planar pitch-linear. YUV420 is always stored as T_L8 with even and odd lines stored to memory as they are received from the CSI interface.

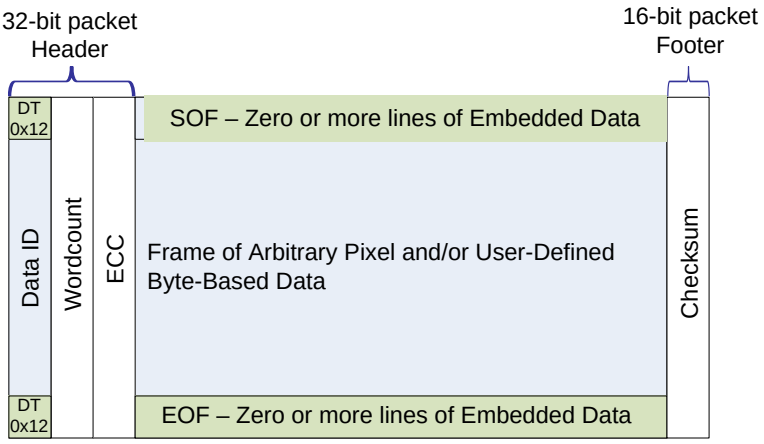
28.2.4.1 YUV Pixel Formatting to ISP Pipeline

The formatting of YUV422 data for presentation to the ISP follows a similar formatting as is done with the raw and RGB formats to convert the data to the unsigned offset binary representation with the exception that an offset of 8192 is used for the chroma channels and an offset of 4096 is used for the luma. The LSB replication follows the same approach as with the RGB and RAW formats. Following the conversion of the luma and chroma to the 14-bit representation for the ISP pipe, the data is then converted from 422 to 444 sampling through interpolation.

28.2.5 Embedded Data Formats

A number of sensor and camera modules utilize embedded data to communicate meta data or calibration data associated with an image capture to the host application. The inclusion of embedded data in the data stream is defined in the MIPI CSI2 specification and has a specific data type of 0x12 assigned. If the embedded data exists, it is identified by the embedded data type at the beginning of a long packet resulting in extra lines added to the beginning or end of a frame. An illustration of embedded and fare data is shown in the following figure.

Figure 112: CSI-2 Incoming Frame Structure with Embedded Data



The CSI2 and VI2 work together to manage the capture and handling of the embedded data during a frame capture. Three basic modes of operation associated with the embedded data capture model are supported. The embedded data capture modes are described in the following section.

28.2.5.1 Embedded Data Capture Modes

The CSI2 can discard embedded data in the CSI2, to pass embedded data through one pixel parser with the image data being presented to the other pixel parser, or to combine the embedded data with the frame data when presented to a single pixel parser.

Embedded data is discarded by setting CSI_PP[A..B]_EMBEDDED_DATA_OPTION to DISCARD in the CSI_PIXEL_STREAM_[A..B]_CONTROL_0 register. If it is desired to keep the embedded data, this field should be set. Depending on the mode of operation and the data format the embedded data, the frame data, or the combined embedded and frame data may be delivered to system memory or the ISP execution resources. The options for embedded data capture are described in the following table.

Table 122: Embedded Data Capture Modes

EMBEDDED	Pixel Format in Memory	Notes
DISCARD	Either PP will deliver frame data only.	MEM or ISPA/ISPB
Embedded Capture with One PP	Embedded Data and Frame Data using Same PP	MEM only
Embedded Capture with Two PPs	One PP used for streaming Embedded Data.	MEM only

EMBEDDED	Pixel Format in Memory	Notes
	One PP used for streaming Frame data.	MEM or ISPA/ISPB

If the mode is set to discard, the embedded data is discarded in the CSI2 and only image data is presented to the VI2 resulting in the normal data processing with the frame data formatting and presentation to memory and/or the ISP. If the mode is set to capture embedded data using one pixel parser, the embedded data is streamed with the image data using the same pixel parser. In this mode the embedded data is combined with the frame data and presented to system memory. The third mode uses both pixel parsers with one pixel parser managing the embedded data and the other managing the frame data.

The discard embedded capture is relatively straightforward and only requires the CSI_PP[A..B]_EMBEDDED_DATA_OPTION to be set to discard with all other programming settings the same as during non-embedded frame capture. The other two modes have some complexity associated with the programming and how the data is organized in memory. The following sections describe the single and dual Pixel Parser (PP) modes in more detail.

Embedded Capture using One PP

The embedded capture using one pixel parser will result in embedded data being stored with the frame data in system memory. Since the line width for the 8-bit embedded data may be longer or shorter than the line with for the pixel data, the line width short/long errors are ignored. In addition, the CSI2 will signal to the VI2 that an incoming line is embedded by setting the CSI[A,B]2VI_FRAME_ERROR during the start of a line. In this fashion, the VI2 will know that the line is embedded and provide for managing the capture and providing the specialized formatting depending on the VI2 memory format setting. The register programming is similar to non-embedded cases except that:

1. VI_CSI_[0..1]_CSI_IMAGE_SIZE register HEIGHT should be programmed with the SOF Embedded Lines+EOF Embedded lines +Normal Lines
2. The CSI_PP[A..B]_EMBEDDED_DATA_OPTION should be set to EMBEDDED in the CSI_PIXEL_STREAM_[A..B]_CONTROL_0 register

In the single PP embedded capture mode, the only destination supported is to MEM since the ISP does not support embedded data with the pixel stream. There are several additional constraints on the supported data formats when using this embedded capture option:

1. RGB Embedded Data capture is not supported.
2. Several Raw flavors with various output options are not supported.
3. YUV embedded capture will combine the embedded data with the Luma plane, and insert zeros into the Chroma planes for some formatting options.

Embedded Capture using Two PPs

The embedded capture using two pixel parsers results in one of the pixel parsers delivering only embedded data to the VI2 and the other delivering only image data. In order to accomplish this, one PP would be programed to parse pixel data and discard embedded data, and the other PP would be programmed to parse only embedded data. The embedded data would then be sent to memory for the VI2 and the pixel data could either be presented to system memory or the ISP.

For example, if an embedded image stream is being captured from CILA, the pixel path could be set up to be CILA → PPA →VI and the embedded data path could be set up to be CILA→PPB→VI→ MEM. The PPA stream programming is similar to normal cases and the stream B programming would be as follows:

1. Route CILA to PPB by setting the CSI_PPB_STREAM_SOURCE to CSI_A
2. Program VI_CSI_1_CSI_IMAGE_SIZE register HEIGHT to be the sum of SOF Embedded Lines+EOF Embedded lines.

3. Program VI_CSI_1_CSI_IMAGE_SIZE register WIDTH to be based on the pixel data line width/data format.
4. Set the DATA_TYPE to EMBED in the VI_CSI_1_CSI_IMAGE_DT register.
5. Set the CSI_PP[A..B]_EMBEDDED_DATA_OPTION to EMBEDDED in the CSI_PIXEL_STREAM_[A..B]_CONTROL_0 register

The programming of the image width in step three is sometimes different than the image width of the normal frame data since the embedded data lines have the same length in bytes as the image data. For example, for a RAW6 64x64 image, there are 48 bytes in each long packet payload resulting in a line width of 48 bytes for the embedded data. Embedded data is always stored as T_L8 in this mode.

28.2.5.2 Frame Data Type with Embedded Data Limitations

The embedded data when using the two pixel parser option is straightforward as long as the frame height and width are programmed appropriate to the data type and the number of embedded lines. The single PP mode of embedded capture is more complex and software will be required to parse the embedded data from the stored embedded data plus image data frame.

The Raw embedded data capture options supported with a single pixel parsers are summarized in the following table.

Table 123: Raw and DPCM Embedded Data Formatting

CSI Format	VI Pixel Format in Memory	1 PP Embed Mode Support	Embedded Data Layout in 32-bit DWORD
RAW6	T_L8	No	NA
RAW7	T_L8	No	NA
RAW8	T_L8	Yes	[emdat3, emdat2, emdat1, emdat0]
RAW10 or DPCM 10-8-10	T_L8	No	NA
	T_R16_I	Yes	*{4'h0,embed_data1,4'hX,4'h0,embed_data0,4'hX}*
	T_X2Lc10Lb10La10	Yes	*{2'b00,embed_data2,2'b00,embed_data1,2'b00,embed_data0}*
RAW12 or DPCM 12-8-12	T_L8	No	NA
	T_R16_I	Yes	*{6'h0,embed_data1,2'hX,6'h0,embed_data0,2'hX}*
RAW14 or DPCM 14-8-14	T_L8	No	NA
	T_R16_I	No	NA

The YUV420 format is very straightforward since only the Luma plane capture is supported. In this capture mode, embedded data is formatted to look like extra rows of equal length as the Luma data since both are packed T_L8. The YUV422 formatting is significantly more complex and some care is necessary to properly parse the data. The YUV embedded data formatting using a single PP is summarized in the following table.

Table 124: YUV Embedded Data Formatting

CSI Format	VI Pixel Format in Memory	1 PP Embed Mode Support	Embedded Data Layout in 32-bit DWORD
YUV420	T_L8	Yes	[emdat3, emdat2, emdat1, emdat0]

CSI Format	VI Pixel Format in Memory	1 PP Embed Mode Support	Embedded Data Layout in 32-bit DWORD
YUV422_8 or YUV422_10	T_L8	Yes	[emdat3, emdat2, emdat1, emdat0]
	T_Y8_U8__Y8_V8	Yes	{8'h0, embed_data1, 8'h0, embed_data0}
	T_U8_Y8__V8_Y8	Yes	{embed_data1, 8'h0, embed_data0, 8'h0}
	T_Y8_V8__Y8_U8	Yes	{8'h0, embed_data1, 8'h0, embed_data0}
	T_V8_Y8__U8_Y8	Yes	{embed_data1, 8'h0, embed_data0, 8'h0}
	T_Y8__U8V8_N422	Yes	*{embed_data3,embed_data2,embed_data1,embed_data0} in Y buffer, The Embedded lines in U/V buffer will be zeroes*
	T_Y8__U8V8_N422	Yes	*{embed_data3,embed_data2,embed_data1,embed_data0} in Y buffer, The Embedded lines in U/V buffer will be zeroes*
	T_Y8__U8__V8_N422	Yes	*{embed_data3,embed_data2,embed_data1,embed_data0} in Y buffer, The Embedded lines in U/V buffer will be zeroes*

28.2.6 Pixel Formats Memory Layout

28.2.6.1 Non-Planar Formats

The bit arrangement of the color components within a pixel for all the supported non-planar color formats are shown in the table below.

Table 125: Pixel Bit Assignments for Non-Planar Formats

Format	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T_L8																									7	6	5	4	3	2	1	0
T_R16_I																	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T_R5G6B5																	4	3	2	1	0	5	4	3	2	1	0	4	3	2	1	0
T_B5G6R5																	4	3	2	1	0	5	4	3	2	1	0	4	3	2	1	0
T_R5G5B5A1																	4	3	2	1	0	4	3	2	1	0	4	3	2	1	0	A 0
T_B5G5R5A1																	4	3	2	1	0	4	3	2	1	0	4	3	2	1	0	A 0
T_A1R5G5B5																	A 0	4	3	2	1	0	4	3	2	1	0	4	3	2	1	0
T_A1B5G5R5																	A 0	4	3	2	1	0	4	3	2	1	0	4	3	2	1	0
T_R4G4B4A4																	3	2	1	0	3	2	1	0	3	2	1	0	A 3	A 2	A 1	A 0
T_B4G4R4A4																	3	2	1	0	3	2	1	0	3	2	1	0	A 3	A 2	A 1	A 0
T_A4R4G4B4																	A 3	A 2	A 1	A 0	3	2	1	0	3	2	1	0	3	2	1	0
T_A4B4G4R4																	A 3	A 2	A 1	A 0	3	2	1	0	3	2	1	0	3	2	1	0
T_R8G8B8A8	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Y 7	Y 6	Y 5	Y 4	Y 3	Y 2	Y 1	Y 0
T_B8G8R8A8	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Y 7	Y 6	Y 5	Y 4	Y 3	Y 2	Y 1	Y 0
T_A8R8G8B8	Y	Y	Y	Y	Y	Y	Y	Y	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Format	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	7	6	5	4	3	2	1	0																								
T_A8B8G8R8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
T_R10G10B10A2	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	A1	A0
T_B10G10R10A2	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	A1	A0
T_A2R10G10B10	A1	A0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
T_A2B10G10R10	A1	A0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
T_V8U8Y8A8	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	A7	A6	A5	A4	A3	A2	A1	A0
T_A8V8U8Y8	A7	A6	A5	A4	A3	A2	A1	A0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
T_Y8_U8__Y8_V8	7	6	5	4	3	2	1	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	7	6	5	4	3	2	1	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
T_Y8_V8__Y8_U8	7	6	5	4	3	2	1	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	7	6	5	4	3	2	1	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
T_U8_Y8__V8_Y8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	7	6	5	4	3	2	1	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	7	6	5	4	3	2	1	0
T_V8_Y8__U8_Y8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	7	6	5	4	3	2	1	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	7	6	5	4	3	2	1	0
T_R16_G16_B16_I	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T_L10__L10__L10_X2			9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0

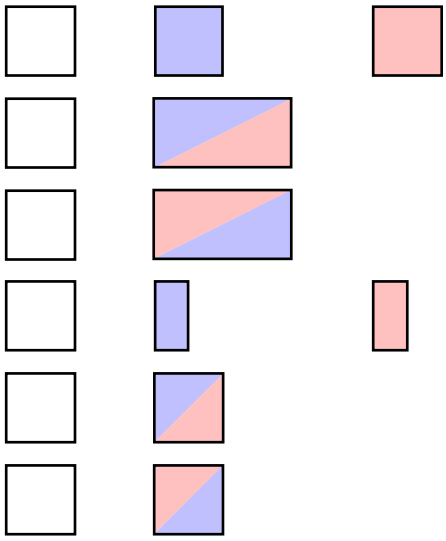
28.2.6.2 Pixel Formats (Planar and Semi-Planar)

Some pixel formats require that the individual color components are separated out into distinct buffers in memory. These formats are defined below:

Table 126: Planar and Semi-Planar Format Definitions

Name	FOURCC	Organization	Sample structure
T_Y8___U8___V8_N444		Planar	4:4:4
T_Y8___U8V8_N444		Semi-planar	4:4:4, U followed by V
T_Y8___V8U8_N444		Semi-planar	4:4:4, V followed by U
T_Y8___U8___V8_N422	YV16	Planar	4:2:2
T_Y8___U8V8_N422	NV16	Semi-planar	4:2:2, U followed by V
T_Y8___V8U8_N422	NV61	Semi-planar	4:2:2, V followed by U

Figure 113: Planar and Semi-Planar Buffer Layout Diagram



28.3 VGP (GPIO) Interface

The VI2 provides for six programmable GPIOs to control various aspects of the camera system. All may be configured as either I/O or PWM and two may also be configured for I2C camera control.

Table 127: VGP1-VGP6 Functions

GPIO	Functions		
	A	B	C
VGP1	I/O	PWM	I2C_SCK
VGP2	I/O	PWM	I2C_SDA
VGP3	I/O	PWM	-
VGP4	I/O	PWM	-
VGP5	I/O	PWM	-
VGP6	I/O	PWM	-

The Pulse Width Modulation (PWM) unit generates a series of 128 identical pulses. The unit can use any of the (VGP1-6) pins as an output, provided that the unit is enabled and the pin is configured to carry PWM data. A pulse's length (in clock cycles), duty cycle, and starting polarity are programmable. A single pulse can take between 2 and 32 clock cycles, where a high or low level is no more than 16 cycles long. A series' length is always 128 times the length of a single pulse. Once the unit is enabled, a series can be either generated once, a set number of times, or repeatedly until the unit is disabled.

28.4 Error Handling

The errors can be broadly classified in the categories provided in the following table.

Table 128: VI Error Categories

Error Categories	Description
CSI Error Indication	D-PHY Level Errors
	Packet Level Errors
	Protocol Decoding Errors
Bad Frame Size	Pixel Per Line Error
	Lines Per Frame Error
Overflow Conditions	Back pressure from ISP
	Backpressure from MCCIF
	Backpressure from ISP in ZSL and SZSL Modes
	Backpressure from MCCIF in ZSL and SZSL Modes
Control Flag Error from CSI	Error in control flag sequence

The rest of this section describes details of various errors and their handling by the VI.

28.4.1 SOF followed by EOF

This is not really an error condition. When the CSI is trying to capture a frame while in single shot mode, if it instead comes across a different frame (not matching the expected parameters of VC/DT/WC), it still sends the SOF indication for the 'bad' frame it dropped and then follows it up with the EOF. This results in the VI receiving an EOF marker without any pixel data or other control flags in between. On receiving this, the VI (depending on the HOSTIF_UPDATE.CAPTURE_GOOD_FRAME configuration) either updates the pending assembly state to active state or overwrites the SOF control flag in the input staging registers.

28.4.2 Missing Control Flags

The missing control flag error occurs when an SOF-SOL-EOL-SOL-EOL-..EOF-SOF sequence occurs. Since the VI does not check for SOL, any error in the sequence concerning SOL is ignored.

28.4.3 Pixels per Line Errors

Pixels Per Line Greater than Image Width

This error is for the case where the first line of the frame was correct and one or more intermediate lines were too long. The CSI will drop any line that does not have the matching WC in the header. This dropping of lines by the CSI might result in a frame-too-short scenario and should be handled as such. The CSI should indicate to the VI when it drops any line via the error flag along with the EOF for that frame. The VI checks for line width and, in case of any error, ignores the rest of the frame.

Pixels Per Line Greater than Image Width

This error is for the case where the first line of the frame was correct and one or more intermediate lines were shorter than expected. This dropping of lines by the CSI might result in a frame-too-short scenario and should be handled as such. The CSI should indicate to VI when it drops any line via the error flag along with the EOF for that frame. The VI checks for line width and, in case of any error, ignores the rest of the frame.

28.4.4 Lines per Frame Errors

Line per Frame Greater than Image Height

In this condition, the input logic should drop the lines beyond the expected frame height and indicate the error to the gatekeeper. On the output side, this error indication is used by output logic to mark the EOF with ‘frame too long’ error logic to ISP. When sending the frame to memory, the VI should drop the additional lines (beyond the expected height) and not corrupt the memory buffer.

Line per Frame Less than Image Height

In this condition, the input logic does not do any special handling. It may mark the EOF with the error flag (not necessarily required as the read logic can identify the frame too short condition too). The input logic still needs to indicate the error to the gatekeeper module for appropriate handling of next set of assembly registers based on CAPTURE_GOOD_FRAME configuration. On the output side of the line-store buffer, the EOF to ISP is marked with frame-too-short error flag.

28.4.5 CSI Error Indication

In this condition, the error flag is used by the gatekeeper module for appropriate handling (update from assembly state or not as defined by the CAPTURE_GOOD_FRAME configuration). The rest of the frame handling follows normal handling or as defined by other defined error-handling scenarios.

28.4.6 Line Buffer Overflow

In case the backpressure from ISP/MC causes the line buffer to overflow, the incoming pixel data from the CSI is dropped until the EOF. On the read side of line-store buffer, if a premature SOF is detected (since the rest of the frame was dropped at the input), the OP*_DONE syncpt is generated as programmed by the CAPTURE_GOOD_FRAME configuration.

28.5 VI Registers

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

28.5.1 VI_CFG_VI_INCR_SYNCPT_0

Offset: 0x0 | Byte Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	VI_COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = VI_MWA_REQ_DONE 5 = VI_MWB_REQ_DONE 6 = VI_MWA_ACK_DONE 7 = VI_MWB_ACK_DONE 8 = VI_ISPA_DONE 9 = VI_CSI_PPA_FRAME_START 10 = VI_CSI_PPB_FRAME_START 11 = VI_CSI_PPA_LINE_START 12 = VI_CSI_PPB_LINE_START 13 = VI_VGP0_RCVD 14 = VI_VGP1_RCVD 15 = VI_ISPB_DONE 16 = COND_16 17 = COND_17 18 = COND_18 19 = COND_19

Bit	Reset	Description
		20 = COND_20 21 = COND_21 22 = COND_22 23 = COND_23 24 = COND_24 25 = COND_25 26 = COND_26 27 = COND_27 28 = COND_28 29 = COND_29 30 = COND_30 31 = COND_31
7:0	0x0	VI_INDX: syncpt index value

28.5.2 VI_CFG_VI_INCR_SYNCPT_CNTRL_0

Offset: 0x1 | Byte Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	VI_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	VI_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.

28.5.3 VI_CFG_VI_INCR_SYNCPT_ERROR_0

Offset: 0x2 | Byte Offset: 0x8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VI_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.

28.5.4 VI_CFG_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).

The 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: 0x8 | Byte Offset: 0x20 | Read/Write: R/W | Reset: 0xXXXXf800 (0bxxxxxxxxxxxxxxxx11111x0000000000)

Bit	R/W	Reset	Description
31:28	RO	X	NEXT_CHANNEL: Next requested channel
25:16	RO	X	NEXT_CLASS: Next requested class
15:12	RW	0xf	CURR_CHANNEL: Current working channel, reset to 'invalid'
11	RW	0x1	AUTO_ACK: Automatically acknowledge any incoming context switch requests 0 = MANUAL 1 = AUTOACK
9:0	RW	0x0	CURR_CLASS: Current working class

28.5.5 VI_CFG_INTSTATUS_0

Offset: 0x9 | Byte Offset: 0x24 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CTXSW_INT: Context switch interrupt status (clear on write)

28.5.6 VI_PWM_CONTROL_0

VI Pulse Width Modulation Control

PWM signal generation logic can generate up to 128 pulses per line internally and the PWM pulse select registers determine 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.

The PWM signal can be output on the VGP6 pin if VGP6 output is enabled and the output select is set to PWM.

The PWM is triggered by the first VSYNC after the PWM_ENABLE bit has been set.

Offset: 0xe | Byte Offset: 0x38 | Read/Write: R/W | Reset: 0x00000000 (0b00000000xx00xxxxxxxxxxxxxxx0xxx0)

Bit	Reset	Description
31:24	0x0	PWM_COUNTER: 8-bit PWM Counter 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 the PWM is turned on, continue through the PWM's 128 cycles repeatedly until the PWM is turned off. Single: After the PWM is turned on, cycle once through the 128 cycles and stop. Counter: After the PWM is turned on, cycle through the 128 cycles PWM_COUNTER number of times then stop. 0 = CONTINUOUS 1 = SINGLE 2 = COUNTER
4	0x0	PWM_DIRECTION: PWM Direction 0= Incrementing 1= Decrementing 0 = INCR 1 = DECR
0	0x0	PWM_ENABLE: PWM Enable 0 = DISABLED 1 = ENABLED

28.5.7 VI_CFG_PWM_HIGH_PULSE_0

PWM High Pulse Period

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

Bit	Reset	Description
31:0	0x0	PWM_HIGH_PULSE: PWM High Pulse

28.5.8 VI_CFG_PWM_LOW_PULSE_0

PWM Low Pulse Period

Offset: 0x10 | Byte Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	PWM_LOW_PULSE: PWM Low Pulse

28.5.9 VI_CFG_PWM_SELECT_PULSE_A_0

PWM Pulse Select A

The next 4 registers select which of the internal 128 pulses to output. Each bit in the four registers corresponds to one internal pulse.

Offset: 0x11 | Byte Offset: 0x44 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PWM_SELECT_A: PWM Select bits 31 to 0

28.5.10 VI_CFG_PWM_SELECT_PULSE_B_0

PWM Pulse Select B

Offset: 0x12 | Byte Offset: 0x48 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PWM_SELECT_B: PWM Select bits 63 to 32

28.5.11 VI_CFG_PWM_SELECT_PULSE_C_0

PWM Pulse Select C

Offset: 0x13 | Byte Offset: 0x4c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PWM_SELECT_C: PWM Select bits 95 to 64

28.5.12 VI_CFG_PWM_SELECT_PULSE_D_0

PWM Pulse Select D

Offset: 0x14 | Byte Offset: 0x50 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PWM_SELECT_D: PWM Select bits 127 to 96

28.5.13 VI_CFG_VGP*n*_0

VI VGP*n* Input/Output Config/Data

There are six VI VGP*n* Config registers, where *n* = 1 through 6.

Offset: 0x19 + (*n* – 1) | Byte Offset: 0x64 + (*n* – 1)*0x4 | Read/Write: R/W | Reset: 0x00000X0X (0b0xxxxxx00xxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
24	RW	0x0	VGP*n*_INPUT_ENABLE: VGP*n* pin Input Enable. This bit controls the VGP*n* pin input. 0 = DISABLED 1 = ENABLED
17	RW	0x0	PIN_OUTPUT_SELECT_VGP*n*: Pin Output Select VGP*n* 0 = VG*Pn*_OUTPUT_DATA 1 = 1'b0
16	RW	0x0	VGP*n*_OUTPUT_ENABLE: VGP*n* pin Output Enable. This bit controls the VGP*n* pin output. 0 = DISABLED 1 = ENABLED
8	RO	X	VGP*n*_INPUT_DATA: VGP*n* pin Input Data (effective if VGP*n*_INPUT_ENABLE is ENABLED) 0= VGP*n* input low 1= VGP*n* input high
0	RW	X	VGP*n*_OUTPUT_DATA: VGP*n* pin Output Data (effective if VGP*n*_OUTPUT_ENABLE is ENABLED and VGP*n*_OUTPUT_SELECT is DATA)

28.5.14 VI_CFG_INTERRUPT_MASK_0

Interrupt Mask

Offset: 0x23 | Byte Offset: 0x8c | Read/Write: R/W | Reset: 0x00000000 (0b000000)

Bit	Reset	Description
6	0x0	VGP6_INT_MASK: VGP6 pin Interrupt Mask. This bit controls interrupting when a VGP6 rising/falling edge is detected. 0 = DISABLED 1 = ENABLED
5	0x0	VGP5_INT_MASK: VGP5 pin Interrupt Mask. This bit controls interrupting when a VGP5 rising/falling edge is detected. 0 = DISABLED 1 = ENABLED
4	0x0	VGP4_INT_MASK: VGP4 pin Interrupt Mask. This bit controls interrupting when a VGP4 rising/falling edge is detected. 0 = DISABLED 1 = ENABLED
3	0x0	VGP3_INT_MASK: VGP3 pin Interrupt Mask. This bit controls interrupting when a VGP3 rising/falling edge is detected. 0 = DISABLED 1 = ENABLED
2	0x0	VGP2_INT_MASK: VGP2 pin Interrupt Mask. This bit controls interrupting when a VGP2 rising/falling edge is detected. 0 = DISABLED 1 = ENABLED
1	0x0	VGP1_INT_MASK: VGP1 pin Interrupt Mask. This bit controls interrupting when a VGP1 rising/falling edge is detected. 0 = DISABLED 1 = ENABLED

28.5.15 VI_CFG_INTERRUPT_TYPE_SELECT_0

Interrupt Type Select

Offset: 0x24 | Byte Offset: 0x90 | Read/Write: R/W | Reset: 0x00000000 (0b000000)

Bit	Reset	Description
6	0x0	VGP6_INT_TYPE: VGP6 pin Interrupt Type. This bit controls interrupt VGP6 0= Edge type 1= Level type 0 = EDGE 1 = LEVEL
5	0x0	VGP5_INT_TYPE: VGP5 pin Interrupt Type. This bit controls interrupt VGP5 0= Edge type 1= Level type 0 = EDGE 1 = LEVEL
4	0x0	VGP4_INT_TYPE: VGP4 pin Interrupt Type. This bit controls interrupt VGP4 0= Edge type 1= Level type 0 = EDGE 1 = LEVEL
3	0x0	VGP3_INT_TYPE: VGP3 pin Interrupt Type. This bit controls interrupt VGP3 0= Edge type 1= Level type 0 = EDGE 1 = LEVEL
2	0x0	VGP2_INT_TYPE: VGP2 pin Interrupt Type. This bit controls interrupt VGP2 0= Edge type 1= Level type 0 = EDGE 1 = LEVEL
1	0x0	VGP1_INT_TYPE: VGP1 pin Interrupt Type. This bit controls interrupt VGP1 0= Edge type 1= Level type 0 = EDGE 1 = LEVEL

28.5.16 VI_CFG_INTERRUPT_POLARITY_SELECT_0

Interrupt Polarity Select

Offset: 0x25 | Byte Offset: 0x94 | Read/Write: R/W | Reset: 0x00000000 (0b000000)

Bit	Reset	Description
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	VGP3_INT_POLARITY: VGP3 pin Interrupt Type. This bit controls interrupt VGP3. 0= falling edge or low level 1= rising edge or high level 0 = LOW 1 = HIGH

Bit	Reset	Description
2	0x0	VGP2_INT_POLARITY: VGP2 pin Interrupt Type. This bit controls interrupt VGP2. 0= falling edge or low level 1= rising edge or high level 0 = LOW 1 = HIGH
1	0x0	VGP1_INT_POLARITY: VGP1 pin Interrupt Type. This bit controls interrupt VGP1. 0= falling edge or low level 1= rising edge or high level 0 = LOW 1 = HIGH

28.5.17 VI_CFG_INTERRUPT_STATUS_0

Interrupt Enable

This register returns interrupt status when read. When this register is written, the interrupt status corresponding to the bits written with 1 is 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 the VI_CFG_INTERRUPT_MASK_0 register, 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: 0x26 | Byte Offset: 0x98 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxx)

Bit	Reset	Description
6	X	VGP6_INT_STATUS: VGP6 pin Interrupt Status. This bit controls interrupt when a 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 a 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 a VGP4 rising/falling edge is detected. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
3	X	VGP3_INT_STATUS: VGP3 pin Interrupt Status. This bit controls interrupt when a VGP3 rising/falling edge is detected. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
2	X	VGP2_INT_STATUS: VGP2 pin Interrupt Status. This bit controls interrupt when a VGP2 rising/falling edge is detected. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
1	X	VGP1_INT_STATUS: VGP1 pin Interrupt Status. This bit controls interrupt when a VGP1 rising/falling

Bit	Reset	Description
		edge is detected. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR

28.5.18 VI_CFG_VGP_SYNCPT_CONFIG_0

Offset: 0x2b | Byte Offset: 0xac | Read/Write: R/W | Reset: 0x00000000 (0b000x000)

Bit	Reset	Description
6:4	0x0	SYNCPT_VGPY_SELECT: Selects the VGP (1-6) for SYNCPT condition VGP_1_RECD
2:0	0x0	SYNCPT_VGPX_SELECT: Selects the VGP (1-6) for SYNCPT condition VGP_0_RECD

28.5.19 VI_CFG_VI_SW_RESET_0

Offset: 0x2d | Byte Offset: 0xb4 | Read/Write: R/W | Reset: 0x00000000 (0b0)

Bit	Reset	Description
0	0x0	MCCIF_RESET: Resets the MCCIF interface

28.5.20 VI_CFG_CG_CTRL_0

Offset: 0x2e | Byte Offset: 0xb8 | Read/Write: R/W | Reset: 0x00000000 (0b0)

Bit	Reset	Description
0	0x0	CG_2ND_LEVEL_EN: Second-level clock gating control 0: Disables any second-level clock gating (MCCIF clocks should be controlled from CAR registers) 1: Enables second-level clock gating 0 = DISABLE 1 = ENABLE

28.5.21 VI_CFG_VI_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control Register (where applicable) and Clock Gating Control

> **Note:** The FIFO timing aspects of this register are not supported but are retained for software compatibility.

The clock override/ovr_mode fields of this register control the 2nd-level clock gating for the client and MC side of the MCCIF. All clock gating is enabled by default.

A '1' written to the rclk/wclk override field will result in one of the following:

- With wclk/rclk override mode = LEGACY, the clock reverts to the legacy mode of operation where the clock is on whenever the client clock is enabled.
- With wclk/rclk override mode = ON, the clock is always on inside the MCCIF and PC.

A '1' written to the cclk override field keeps the client's clock always on inside the MCCIF.

Offset: 0x39 | Byte Offset: 0xe4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx00000xxxxxxxxxxxxxx00)

Bit	Reset	Description
20	LEGACY	VI_RCLK_OVR_MODE: 0 = LEGACY 1 = ON

Bit	Reset	Description
19	LEGACY	VI_WCLK_OVR_MODE: 0 = LEGACY 1 = ON
18	0x0	VI_CCLK_OVERRIDE
17	0x0	VI_RCLK_OVERRIDE
16	0x0	VI_WCLK_OVERRIDE
1	DISABLE	VI_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	VI_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

28.5.22 VI_CFG_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: 0x3a | Byte Offset: 0xe8 | Read/Write: R/W | Reset: 0x00000032 (0b00110010)

Bit	Reset	Description
7:0	0x32	VIWSB_WCOAL_TMVAL

28.5.23 VI_CFG_DVFS_0

Dynamic Voltage Frequency Shift Register

Offset: 0x3c | Byte Offset: 0xf0 | Read/Write: R/W | Reset: 0x4040007f (0b1000000x1000000xxxxxxxxx1111111)

Bit	Reset	Description
30:24	0x40	SENSORB_LB_THRESHOLD: This field defines the DVFS threshold for the VI Line buffer for Sensor B, which is compared against the buffer utilization in order to assert the ready_for_latency event signal to the MC. The final ready_for_latency_signal is ANDed with the one generated from the MCCIF_THRESHOLD comparison.
22:16	0x40	SENSORA_LB_THRESHOLD: This field defines the DVFS threshold for the VI Line buffer for Sensor A, which is compared against the buffer utilization in order to assert the ready_for_latency event signal to the MC. The final ready_for_latency_signal is ANDed with the one generated from the MCCIF_THRESHOLD comparison.
6:0	0x7f	MCCIF_THRESHOLD: This field defines the DVFS threshold for the MCCIF FIFO, which is compared against the write request buffer utilization in order to assert the ready_for_latency_event signal to the MC.

28.5.24 VI_CFG_RESERVE_0_0

Reserved register

Offset: 0x3d | Byte Offset: 0xf4 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:12	X	nc_RESERVE_0_3
11:8	X	nc_RESERVE_0_2

Bit	Reset	Description
7:4	X	nc_RESERVE_0_1
3:0	X	nc_RESERVE_0_0

28.5.25 VI_CFG_RESERVE_1_0

Reserved register

Offset: 0x3e | Byte Offset: 0xf8 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:12	X	nc_RESERVE_1_3
11:8	X	nc_RESERVE_1_2
7:4	X	nc_RESERVE_1_1
3:0	X	nc_RESERVE_1_0

28.6 VI Input CSI Interface Registers

CSI Channel Registers

28.6.1 VI_CSI_0_SW_RESET_0

Offset: 0x40 | Byte Offset: 0x100 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	ISPINTF_RESET: Reset ISP interface
3	0x0	MCINTF_RESET: Reset Memory Client i/f logic
2	0x0	PF_RESET: Reset Pixel format logic
1	0x0	SENSORCTL_RESET: Reset Sensor control logic
0	0x0	SHADOW_RESET: Reset Shadow copy logic

28.6.2 VI_CSI_0_SINGLE_SHOT_0

Offset: 0x41 | Byte Offset: 0x104 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CAPTURE: request update of the previously assembled 'assembly state' to 'active state' and schedule single shot capture for the VI channel. The actual update of operational register is gated by the next EOF Register read back returns pending capture status (at input of VI)

28.6.3 VI_CSI_0_SINGLE_SHOT_STATE_UPDATE_0

Offset: 0x42 | Byte Offset: 0x108 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	CAPTURE_GOOD_FRAME: 1'b0: update state when the next SOF is received, irrespective of the state of previous frame. 1'b1: update state when the next SOF is received AND the previous frame was the correct frame based on current operational state.

28.6.4 VI_CSI_0_IMAGE_DEF_0

Offset: 0x43 | Byte Offset: 0x10c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000000 (0bxxxxxxx000000000xxxxxxx0xxxxx000)

Bit	Reset	Description
24	0x0	BYPASS_PXL_TRANSFORM: Bypass pixel transformation VI unit does convert the pixels into I.F format while interfacing with Memory packer. This bit can be used to bypass this conversion and will useful in writing compressed image format into memory.
23:16	0x0	FORMAT: Pixel memory format for the VI channel In the enums below, the following are not supported in Tegra K1: T_A2Y10U10V10, T_V10U10Y10A2 T_Y8__U8__V8_N444, T_Y8__U8V8_N444, T_Y8__V8U8_N444 16 = T_L8 32 = T_R16_I 33 = T_B5G6R5 34 = T_R5G6B5 35 = T_A1B5G5R5 36 = T_A1R5G5B5 37 = T_B5G5R5A1 38 = T_R5G5B5A1 39 = T_A4B4G4R4 40 = T_A4R4G4B4 41 = T_B4G4R4A4 42 = T_R4G4B4A4 64 = T_A8B8G8R8 65 = T_A8R8G8B8 66 = T_B8G8R8A8 67 = T_R8G8B8A8 68 = T_A2B10G10R10 69 = T_A2R10G10B10 70 = T_B10G10R10A2 71 = T_R10G10B10A2 193 = T_A8Y8U8V8 194 = T_V8U8Y8A8 197 = T_A2Y10U10V10 198 = T_V10U10Y10A2 200 = T_Y8_U8__Y8_V8 201 = T_Y8_V8__Y8_U8 202 = T_U8_Y8__V8_Y8 203 = T_V8_Y8__U8_Y8 224 = T_Y8__U8__V8_N444 225 = T_Y8__U8V8_N444 226 = T_Y8__V8U8_N444 227 = T_Y8__U8__V8_N422 228 = T_Y8__U8V8_N422 229 = T_Y8__V8U8_N422 230 = T_Y8__U8__V8_N420 231 = T_Y8__U8V8_N420 232 = T_Y8__V8U8_N420 233 = T_X2Lc10Lb10La10 234 = T_A2R6R6R6R6R6
8	0x0	INTERLEAVING_MODE: 1= Enable interleaving mode for writing image into memory
2	0x0	DEST_ISPB: 1= send VI channel data to ISPB
1	0x0	DEST_ISPA: 1= send VI channel data to ISPA
0	0x0	DEST_MEM: 1= send VI channel data to MEM

28.6.5 VI_CSI_0_RGB2Y_CTRL_0

Offset: 0x44 | Byte Offset: 0x110 | Read/Write: R/W | Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:18	X	B2Y_COEFF: Blue coefficient used to convert RGB pixel data to Y for writing to Memory (in U0.6

Bit	Reset	Description
		format)
15:10	X	G2Y_COEFF: Green coefficient used to convert RGB pixel data to Y for writing to Memory (in U0.6 format)
7:2	X	R2Y_COEFF: Red coefficient used to convert RGB pixel data to Y for writing to Memory (in U0.6 format)

28.6.6 VI_CSI_0_MEM_TILING_0

Offset: 0x45 | Byte Offset: 0x114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	TILING_FORMAT: VI channel memory surface tiling format 254 = BLOCK_LINEAR 0 = PITCH_LINEAR

28.6.7 VI_CSI_0_CSI_IMAGE_SIZE_0

Offset: 0x46 | Byte Offset: 0x118 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	HEIGHT: Height of VI channel frame in Lines. This Register field is double buffered.
15:0	X	WIDTH: Width of VI channel frame in Pixels. This Register field is double buffered.

28.6.8 VI_CSI_0_CSI_IMAGE_SIZE_WC_0

Offset: 0x47 | Byte Offset: 0x11c | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	WORDCOUNT: VI channel word count for the Image to be captured This parameter specifies the number of bytes per line/packet. This is matched against the Word Count field in packet header if enabled. 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 ---------- This register is used in to set the wordcount in Test Pattern Generation mode. The wordcount size needed to be set in 16-pixel boundary depended on data type selection (RAW10 or RGB888) This Register field is double buffered.

28.6.9 VI_CSI_0_CSI_IMAGE_DT_0

Offset: 0x48 | Byte Offset: 0x120 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxx0xxxxxxxxxxx)

Bit	Reset	Description
12	0x0	INTERLACED_VIDEO: VI channel interlaced video format enable
9:8	X	VC: VI channel virtual channel ID. This Register field is double buffered.
5:0	X	DATA_TYPE: VI channel input data type. This Register field is double buffered. 18 = EMBED 24 = YUV420_8 25 = YUV420_10 26 = LEG_YUV420_8

Bit	Reset	Description
		28 = YUV420CSPS_8 29 = YUV420CSPS_10 30 = YUV422_8 31 = YUV422_10 32 = RGB444 33 = RGB555 34 = RGB565 35 = RGB666 36 = RGB888 40 = RAW6 41 = RAW7 42 = RAW8 43 = RAW10 44 = RAW12 45 = RAW14 48 = ARB_DT1 49 = ARB_DT2 50 = ARB_DT3 51 = ARB_DT4

28.6.10 VI_CSI_0_SURFACE0_OFFSET_MSB_0

Offset: 0x49 | Byte Offset: 0x124 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	OFFSET_MSB_0: Base address MSB for Surface 0 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered.

28.6.11 VI_CSI_0_SURFACE0_OFFSET_LSB_0

Offset: 0x4a | Byte Offset: 0x128 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	OFFSET_LSB_0: Base address LSB for Surface 0 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered.

28.6.12 VI_CSI_0_SURFACE1_OFFSET_MSB_0

Offset: 0x4b | Byte Offset: 0x12c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	OFFSET_MSB_1: Base address MSB for Surface 1 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered.

28.6.13 VI_CSI_0_SURFACE1_OFFSET_LSB_0

Offset: 0x4c | Byte Offset: 0x130 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	OFFSET_LSB_1: Base address LSB for Surface 1 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered.

28.6.14 VI_CSI_0_SURFACE2_OFFSET_MSB_0

Offset: 0x4d | Byte Offset: 0x134 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	OFFSET_MSB_2: Base address MSB for Surface 2 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered.

28.6.15 VI_CSI_0_SURFACE2_OFFSET_LSB_0

Offset: 0x4e | Byte Offset: 0x138 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	OFFSET_LSB_2: Base address LSB for Surface 2 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered.

28.6.16 VI_CSI_0_SURFACE0_BF_OFFSET_MSB_0

Offset: 0x4f | Byte Offset: 0x13c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	BF_OFFSET_MSB_0: Base address MSB for Bottom Frame surface 0 in Memory (for Interlaced Video). This Register field is double buffered.

28.6.17 VI_CSI_0_SURFACE0_BF_OFFSET_LSB_0

Offset: 0x50 | Byte Offset: 0x140 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BF_OFFSET_LSB_0: Base address LSB for Bottom Frame surface 0 in Memory (for Interlaced Video). This Register field is double buffered.

28.6.18 VI_CSI_0_SURFACE1_BF_OFFSET_MSB_0

Offset: 0x51 | Byte Offset: 0x144 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	BF_OFFSET_MSB_1: Base address MSB for Bottom Frame surface 1 in Memory (for Interlaced Video). This Register field is double buffered.

28.6.19 VI_CSI_0_SURFACE1_BF_OFFSET_LSB_0

Offset: 0x52 | Byte Offset: 0x148 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BF_OFFSET_LSB_1: Base address LSB for Bottom Frame surface 1 in Memory (for Interlaced Video). This Register field is double buffered.

28.6.20 VI_CSI_0_SURFACE2_BF_OFFSET_MSB_0

Offset: 0x53 | Byte Offset: 0x14c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	BF_OFFSET_MSB_2: Base address MSB for Bottom Frame surface 2 in Memory (for Interlaced Video). This Register field is double buffered.

28.6.21 VI_CSI_0_SURFACE2_BF_OFFSET_LSB_0

Offset: 0x54 | Byte Offset: 0x150 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BF_OFFSET_LSB_2: Base address LSB for Bottom Frame surface 2 in Memory (for Interlaced Video). This Register field is double buffered.

28.6.22 VI_CSI_0_SURFACE0_STRIDE_0

Offset: 0x55 | Byte Offset: 0x154 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	STRIDE_0: Stride for Surface 0 in Memory. This Register field is double buffered.

28.6.23 VI_CSI_0_SURFACE1_STRIDE_0

Offset: 0x56 | Byte Offset: 0x158 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	STRIDE_1: Stride for Surface 1 in Memory. This Register field is double buffered.

28.6.24 VI_CSI_0_SURFACE2_STRIDE_0

Offset: 0x57 | Byte Offset: 0x15c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	STRIDE_2: Stride for Surface 2 in Memory This Register field is double buffered.

28.6.25 VI_CSI_0_SURFACE_HEIGHT0_0

Offset: 0x58 | Byte Offset: 0x160 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0001)

Bit	Reset	Description
3:0	0x1	GOBS: Number of GOBs stacked verically to form a block

28.6.26 VI_CSI_0_ISPINTF_CONFIG_0

Offset: 0x59 | Byte Offset: 0x164 | Read/Write: R/W | Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx011)

Bit	Reset	Description
2:1	0x1	CHROMA_POS: Stream format 0 = MPEG1 1 = MPEG2
0	0x1	DO_YUV_INTERP: DO_YUV_INTERP: 0=Bypass Chroma Interpolator for YUV 1=Run YUV through Chroma Interpolator

28.6.27 VI_CSI_0_ERROR_STATUS_0

Offset: 0x61 | Byte Offset: 0x184 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
5	X	WATCHDOG_INT: This event occurs when Watchdog timer expires and the EOF is not received on the CSI2VI interface
4	X	CSI_FRAME_ERROR: Flagged if EOF field from CSI has FRAME_ERROR bit set
3	X	FRAME_HEIGHT_LONG_ERROR: Flagged if frame height is larger than HEIGHT. Write 1 to clear.
2	X	FRAME_HEIGHT_SHORT_ERROR: Flagged if frame height is smaller than HEIGHT. Write 1 to clear.
1	X	LINE_WIDTH_LONG_ERROR: Flagged if frame line width is larger than WIDTH. Write 1 to clear.
0	X	LINE_WIDTH_SHORT_ERROR: Flagged if frame line width is smaller than WIDTH. Write 1 to clear.

28.6.28 VI_CSI_0_ERROR_INT_MASK_0

Offset: 0x62 | Byte Offset: 0x188 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5	0x0	WATCHDOG_INT_MASK: Watchdog Timer 0 Interrupt Mask This controls interrupt when WATCHDOG trigger event occurs for the CSI stream 0 = Disabled, 1 = Enabled
4	0x0	CSI_FRAME_ERROR_INT_MASK: CSI error interrupt mask
3	0x0	FRAME_HEIGHT_LONG_INT_MASK: frame height long error interrupt mask
2	0x0	FRAME_HEIGHT_SHORT_INT_MASK: frame height short error interrupt mask
1	0x0	LINE_WIDTH_LONG_INT_MASK: line width long error interrupt mask
0	0x0	LINE_WIDTH_SHORT_INT_MASK: line width short error interrupt mask

28.6.29 VI_CSI_0_WD_CTRL_0

Watch Dog Timer Control Register

Offset: 0x63 | Byte Offset: 0x18c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	WD_ENABLE: Watch Dog Timer Enable 0 = DISABLE 1 = ENABLE

28.6.30 VI_CSI_0_WD_PERIOD_0

Watch Dog Timer Control Register

Offset: 0x64 | Byte Offset: 0x190 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	WD_PERIOD: Watch Dog Timer Period in pixel clocks

28.6.31 VI_CSI_1_SW_RESET_0

Offset: 0x80 | Byte Offset: 0x200 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	ISPINTF_RESET: Reset ISP interface
3	0x0	MCINTF_RESET: Reset Memory Client i/f logic
2	0x0	PF_RESET: Reset Pixel format logic
1	0x0	SENSORCTL_RESET: Reset Sensor control logic
0	0x0	SHADOW_RESET: Reset Shadow copy logic

28.6.32 VI_CSI_1_SINGLE_SHOT_0

Offset: 0x81 | Byte Offset: 0x204 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CAPTURE: request update of the previously assembled 'assembly state' to 'active state' and schedule single shot capture for the VI channel. The actual update of operational register is gated by the next EOF Register read back returns pending capture status (at input of VI)

28.6.33 VI_CSI_1_SINGLE_SHOT_STATE_UPDATE_0

Offset: 0x82 | Byte Offset: 0x208 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	CAPTURE_GOOD_FRAME: 1'b0: update state when the next SOF is received, irrespective of the state of previous frame. 1'b1: update state when the next SOF is received AND the previous frame was the correct frame based on current operational state.

28.6.34 VI_CSI_1_IMAGE_DEF_0

Offset: 0x83 | Byte Offset: 0x20c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000000 (0bxxxxxxx000000000xxxxxxx0xxxxx000)

Bit	Reset	Description
24	0x0	BYPASS_PXL_TRANSFORM: Bypass pixel transformation VI unit does convert the pixels into I.F format while interfacing with Memory packer. This bit can be used to bypass this convertion and will useful in writing compressed image format into memory.
23:16	0x0	FORMAT: Pixel memory format for the VI channel In the enum below, the following are not supported in Tegra K1: T_A2Y10U10V10, T_V10U10Y10A2 T_Y8__U8__V8_N444, T_Y8__U8V8_N444, T_Y8__V8U8_N444 16 = T_L8 32 = T_R16_I 33 = T_B5G6R5 34 = T_R5G6B5 35 = T_A1B5G5R5 36 = T_A1R5G5B5 37 = T_B5G5R5A1 38 = T_R5G5B5A1 39 = T_A4B4G4R4 40 = T_A4R4G4B4 41 = T_B4G4R4A4

Bit	Reset	Description
		42 = T_R4G4B4A4 64 = T_A8B8G8R8 65 = T_A8R8G8B8 66 = T_B8G8R8A8 67 = T_R8G8B8A8 68 = T_A2B10G10R10 69 = T_A2R10G10B10 70 = T_B10G10R10A2 71 = T_R10G10B10A2 193 = T_A8Y8U8V8 194 = T_V8U8Y8A8 197 = T_A2Y10U10V10 198 = T_V10U10Y10A2 200 = T_Y8_U8__Y8_V8 201 = T_Y8_V8__Y8_U8 202 = T_U8_Y8__V8_Y8 203 = T_V8_Y8__U8_Y8 224 = T_Y8__U8__V8_N444 225 = T_Y8__U8V8_N444 226 = T_Y8__V8U8_N444 227 = T_Y8__U8__V8_N422 228 = T_Y8__U8V8_N422 229 = T_Y8__V8U8_N422 230 = T_Y8__U8__V8_N420 231 = T_Y8__U8V8_N420 232 = T_Y8__V8U8_N420 233 = T_X2Lc10Lb10La10 234 = T_A2R6R6R6R6R6
8	0x0	INTERLEAVING_MODE: 1= Enable interleaving mode for writing image into memory
2	0x0	DEST_ISPB: 1= send VI channel data to ISPB
1	0x0	DEST_ISPA: 1= send VI channel data to ISPA
0	0x0	DEST_MEM: 1= send VI channel data to MEM

28.6.35 VI_CSI_1_RGB2Y_CTRL_0

Offset: 0x84 | Byte Offset: 0x210 | Read/Write: R/W | Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:18	X	B2Y_COEFF: Blue coefficient used to convert RGB pixel data to Y for writing to Memory(in U0.6 format)
15:10	X	G2Y_COEFF: Green coefficient used to convert RGB pixel data to Y for writing to Memory(in U0.6 format)
7:2	X	R2Y_COEFF: Red coefficient used to convert RGB pixel data to Y for writing to Memory(in U0.6 format)

28.6.36 VI_CSI_1_MEM_TILING_0

Offset: 0x85 | Byte Offset: 0x214 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	TILING_FORMAT: VI channel memory surface tiling format 254 = BLOCK_LINEAR 0 = PITCH_LINEAR

28.6.37 VI_CSI_1_CSI_IMAGE_SIZE_0

Offset: 0x86 | Byte Offset: 0x218 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	HEIGHT: Height of VI channel frame in Lines. This Register field is double buffered.
15:0	X	WIDTH: Width of VI channel frame in Pixels. This Register field is double buffered.

28.6.38 VI_CSI_1_CSI_IMAGE_SIZE_WC_0

Offset: 0x87 | Byte Offset: 0x21c | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	WORDCOUNT: VI channel word count for the Image to be captured This parameter specifies the number of bytes per line/packet. This is matched against the Word Count field in packet header if enabled. 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 ---------- This register is used in to set the wordcount in Test Pattern Generation mode. The wordcount size needed to be set in 16-pixel boundary depended on data type selection (RAW10 or RGB888) This Register field is double buffered.

28.6.39 VI_CSI_1_CSI_IMAGE_DT_0

Offset: 0x88 | Byte Offset: 0x220 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxx0xxxxxxxxxxx)

Bit	Reset	Description
12	0x0	INTERLACED_VIDEO: VI channel interlaced video format enable
9:8	X	VC: VI channel virtual channel ID. This Register field is double buffered.
5:0	X	DATA_TYPE: VI channel input data type. This Register field is double buffered. 18 = EMBED 24 = YUV420_8 25 = YUV420_10 26 = LEG_YUV420_8 28 = YUV420CSPS_8 29 = YUV420CSPS_10 30 = YUV422_8 31 = YUV422_10 32 = RGB444 33 = RGB555 34 = RGB565 35 = RGB666 36 = RGB888 40 = RAW6 41 = RAW7 42 = RAW8 43 = RAW10 44 = RAW12 45 = RAW14 48 = ARB_DT1 49 = ARB_DT2 50 = ARB_DT3 51 = ARB_DT4

28.6.40 VI_CSI_1_SURFACE0_OFFSET_MSB_0

Offset: 0x89 | Byte Offset: 0x224 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	OFFSET_MSB_0: Base address MSB for Surface 0 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered.

28.6.41 VI_CSI_1_SURFACE0_OFFSET_LSB_0

Offset: 0x8a | Byte Offset: 0x228 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	OFFSET_LSB_0: Base address LSB for Surface 0 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered.

28.6.42 VI_CSI_1_SURFACE1_OFFSET_MSB_0

Offset: 0x8b | Byte Offset: 0x22c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	OFFSET_MSB_1: Base address MSB for Surface 1 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered.

28.6.43 VI_CSI_1_SURFACE1_OFFSET_LSB_0

Offset: 0x8c | Byte Offset: 0x230 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	OFFSET_LSB_1: Base address LSB for Surface 1 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered.

28.6.44 VI_CSI_1_SURFACE2_OFFSET_MSB_0

Offset: 0x8d | Byte Offset: 0x234 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	OFFSET_MSB_2: Base address MSB for Surface 2 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered.

28.6.45 VI_CSI_1_SURFACE2_OFFSET_LSB_0

Offset: 0x8e | Byte Offset: 0x238 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	OFFSET_LSB_2: Base address LSB for Surface 2 in Memory (or Top Frame for Interlaced Video) This Register field is double buffered.

28.6.46 VI_CSI_1_SURFACE0_BF_OFFSET_MSB_0

Offset: 0x8f | Byte Offset: 0x23c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	BF_OFFSET_MSB_0: Base address MSB for Bottom Frame surface 0 in Memory (for Interlaced Video) This Register field is double buffered.

28.6.47 VI_CSI_1_SURFACE0_BF_OFFSET_LSB_0

Offset: 0x90 | Byte Offset: 0x240 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BF_OFFSET_LSB_0: Base address LSB for Bottom Frame surface 0 in Memory (for Interlaced Video) This Register field is double buffered.

28.6.48 VI_CSI_1_SURFACE1_BF_OFFSET_MSB_0

Offset: 0x91 | Byte Offset: 0x244 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	BF_OFFSET_MSB_1: Base address MSB for Bottom Frame surface 1 in Memory (for Interlaced Video) This Register field is double buffered.

28.6.49 VI_CSI_1_SURFACE1_BF_OFFSET_LSB_0

Offset: 0x92 | Byte Offset: 0x248 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BF_OFFSET_LSB_1: Base address LSB for Bottom Frame surface 1 in Memory (for Interlaced Video) This Register field is double buffered.

28.6.50 VI_CSI_1_SURFACE2_BF_OFFSET_MSB_0

Offset: 0x93 | Byte Offset: 0x24c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	BF_OFFSET_MSB_2: Base address MSB for Bottom Frame surface 2 in Memory (for Interlaced Video) This Register field is double buffered.

28.6.51 VI_CSI_1_SURFACE2_BF_OFFSET_LSB_0

Offset: 0x94 | Byte Offset: 0x250 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BF_OFFSET_LSB_2: Base address LSB for Bottom Frame surface 2 in Memory (for Interlaced Video) This Register field is double buffered.

28.6.52 VI_CSI_1_SURFACE0_STRIDE_0

Offset: 0x95 | Byte Offset: 0x254 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	STRIDE_0: Stride for Surface 0 in Memory This Register field is double buffered.

28.6.53 VI_CSI_1_SURFACE1_STRIDE_0

Offset: 0x96 | Byte Offset: 0x258 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	STRIDE_1: Stride for Surface 1 in Memory This Register field is double buffered.

28.6.54 VI_CSI_1_SURFACE2_STRIDE_0

Offset: 0x97 | Byte Offset: 0x25c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	STRIDE_2: Stride for Surface 2 in Memory This Register field is double buffered.

28.6.55 VI_CSI_1_SURFACE_HEIGHT0_0

Offset: 0x98 | Byte Offset: 0x260 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0001)

Bit	Reset	Description
3:0	0x1	GOBS: Number of GOBs stacked verically to form a block

28.6.56 VI_CSI_1_ISPINTF_CONFIG_0

Offset: 0x99 | Byte Offset: 0x264 | Read/Write: R/W | Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx011)

Bit	Reset	Description
2:1	0x1	CHROMA_POS: Stream format 0 = MPEG1 1 = MPEG2
0	0x1	DO_YUV_INTERP: DO_YUV_INTERP: 0=Bypass Chroma Interpolator for YUV 1=Run YUV through Chroma Interpolator

28.6.57 VI_CSI_1_ERROR_STATUS_0

Offset: 0xa1 | Byte Offset: 0x284 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
5	X	WATCHDOG_INT: This event occurs when Watchdog timer expires and the EOF is not received on the CSI2VI interface
4	X	CSI_FRAME_ERROR: Flagged if EOF field from CSI has FRAME_ERROR bit set
3	X	FRAME_HEIGHT_LONG_ERROR: Flagged if frame height is larger than HEIGHT. Write 1 to clear.
2	X	FRAME_HEIGHT_SHORT_ERROR: Flagged if frame height is smaller than HEIGHT. Write 1 to clear.
1	X	LINE_WIDTH_LONG_ERROR: Flagged if frame line width is larger than WIDTH. Write 1 to clear.
0	X	LINE_WIDTH_SHORT_ERROR: Flagged if frame line width is smaller than WIDTH. Write 1 to clear.

28.6.58 VI_CSI_1_ERROR_INT_MASK_0

Offset: 0xa2 | Byte Offset: 0x288 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5	0x0	WATCHDOG_INT_MASK: Watchdog Timer 0 Interrupt Mask This controls interrupt when WATCHDOG trigger event occurs for the CSI stream 0 = Disabled, 1 = Enabled
4	0x0	CSI_FRAME_ERROR_INT_MASK: CSI error interrupt mask
3	0x0	FRAME_HEIGHT_LONG_INT_MASK: frame height long error interrupt mask
2	0x0	FRAME_HEIGHT_SHORT_INT_MASK: frame height short error interrupt mask
1	0x0	LINE_WIDTH_LONG_INT_MASK: line width long error interrupt mask
0	0x0	LINE_WIDTH_SHORT_INT_MASK: line width short error interrupt mask

28.6.59 VI_CSI_1_WD_CTRL_0

Watch Dog Timer Control Register

Offset: 0xa3 | Byte Offset: 0x28c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	WD_ENABLE: Watch Dog Timer Enable 0 = DISABLE 1 = ENABLE

28.6.60 VI_CSI_1_WD_PERIOD_0

Watch Dog Timer Control Register

Offset: 0xa4 | Byte Offset: 0x290 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	WD_PERIOD: Watch Dog Timer Period in pixel clocks

28.7 MIPI-CSI Registers

Refer to the CSI section in this document for descriptions of these registers.

[THIS PAGE INTENTIONALLY LEFT BLANK]

29.0 SD/MMC CONTROLLER

The SD/MMC Controller can interface with SD/eSD, SDIO, or eMMC devices. It supports both on-board devices and plug-in cards.

Each Tegra® K1 mobile processor contains four instances of this controller, each of which is identical in its internal function, but they vary in I/O capabilities:

Controller	UHS-I Signaling Supported	8-bit eMMC Supported	Signaling Voltages
SDMMC1	Yes	No	1.8, 2.8-3.3 V
SDMMC2	Yes	Yes	1.8, 2.8-3.3 V
SDMMC3	Yes	No	1.8, 2.8-3.3 V
SDMMC4	Yes	Yes	1.2, 1.8 V

29.1 Supported Specifications and Standards

The SD/MMC controller supports the specifications from the SD Card Association and JEDEC (MMC) at the versions listed in the following table.

Protocol	Version
SD	4.0 (UHS-I only)
SDIO	4.0 (UHS-I only)
eSD	2.1
eMMC	4.51
SDHOST	4.0 (UHS-I only)

The specifications are as follows:

- “SD Specifications, Part E1, SDIO Specification”, Technical Committee SD Card Association, Version 4.00, 20-February 2012
- “SD Specifications, Part 1, Physical Layer Specification”, Technical Committee SD Card Association, Version 4.00, 30-May 2011.
- “SD Specification, Part 1, eSD (Embedded SD) Addendum”, Technical Committee SD Card Association, Version 2.10, 25-November 2008.
- “SD Specifications, Part A2, SD Host Controller Standard Specification”, Technical Committee SD Card Association, Version 4.00, 20-February 2012.
- “JEDEC Standard, Embedded Multimedia Card (eMMC) Electrical Standard 4.51”, JEDEC Solid State Technology Association, June 2012.

29.2 Supported Speeds

These tables show the maximum data rates available over the physical interface. The speeds achievable in actual use will be less than these rates, depending on the performance of the device itself, protocol limitations, and the software driver.

For SD 3.0 data transfer modes, these are the maximum data transfer speeds:

Speed Mode	Signal Voltage	I/O Frequency (MHz)	Bus Width	Maximum Throughput (MBytes/s)	UHS Protocol
Default Speed	3.3	25	1,4	12.5	UHS-I, 50 (SD3.0) UHS-I, 104 (SD3.0)
High Speed	3.3	50	1,4	25	UHS-I, 50 (SD3.0) UHS-I, 104 (SD3.0)
SDR12	1.8	25	1,4	12.5	UHS-I, 50 (SD3.0) UHS-I, 104 (SD3.0)
SDR25	1.8	50	1,4	25	UHS-I, 50 (SD3.0) UHS-I, 104 (SD3.0)
SDR50	1.8	100	1,4	50	UHS-I, 50 (SD3.0) UHS-I, 104 (SD3.0)
SDR104 UHS-I	1.8	208	1,4	104	UHS-I, 104 (SD3.0)

For eSD 2.1 data transfer modes, these are the maximum data transfer speeds:

Speed Mode	Signal Voltage	I/O Frequency (MHz)	Bus Width	Maximum Throughput (MBytes/s)	eSD Protocol
Default Speed	3.3	25	1,4,8	12.5	eSD 2.1
High Speed	3.3	50	1,4,8	25	eSD 2.1

For SDIO data transfer modes, these are the maximum data transfer speeds:

Speed Mode	I/O Frequency (MHz)	Bus Width	Maximum Throughput (MBytes/s)	Protocol Version/UHS Version
Default Speed	25	1,4	12.5	SDIO2.0 /UHS-I, 50/ UHS-I, 104
High Speed	50	1,4	25	SDIO2.0 /UHS-I, 50/ UHS-I, 104
SDR12	25	1,4	12.5	UHS-I, 50 (SDIO3.0) UHS-I, 104 (SDIO3.0)
SDR25	50	1,4	25	UHS-I, 50 (SDIO3.0) UHS-I, 104 (SDIO3.0)
SDR50	100	1,4	50	UHS-I, 50 (SDIO3.0) UHS-I, 104 (SDIO3.0)
SDR104 UHS-I	208	1,4	104	UHS-I, 104 (SDIO3.0)

For eMMC data transfer modes, these are the maximum data transfer speeds:

Speed Mode	I/O Frequency (MHz)	Bus Width	Maximum Throughput (MBytes/s)	MMC Revision
Legacy Speed	26	1,4,8	26	MMC 4.3
High Speed SDR	52	1,4,8	52	MMC 4.3
High Speed DDR	52	4,8	104	MMC 4.4

Speed Mode	I/O Frequency (MHz)	Bus Width	Maximum Throughput (MBytes/s)	MMC Revision
HS200 (SDR)	200	4,8	200	MMC 4.5

29.3 Operation

29.3.1 Hardware / Software Partitioning

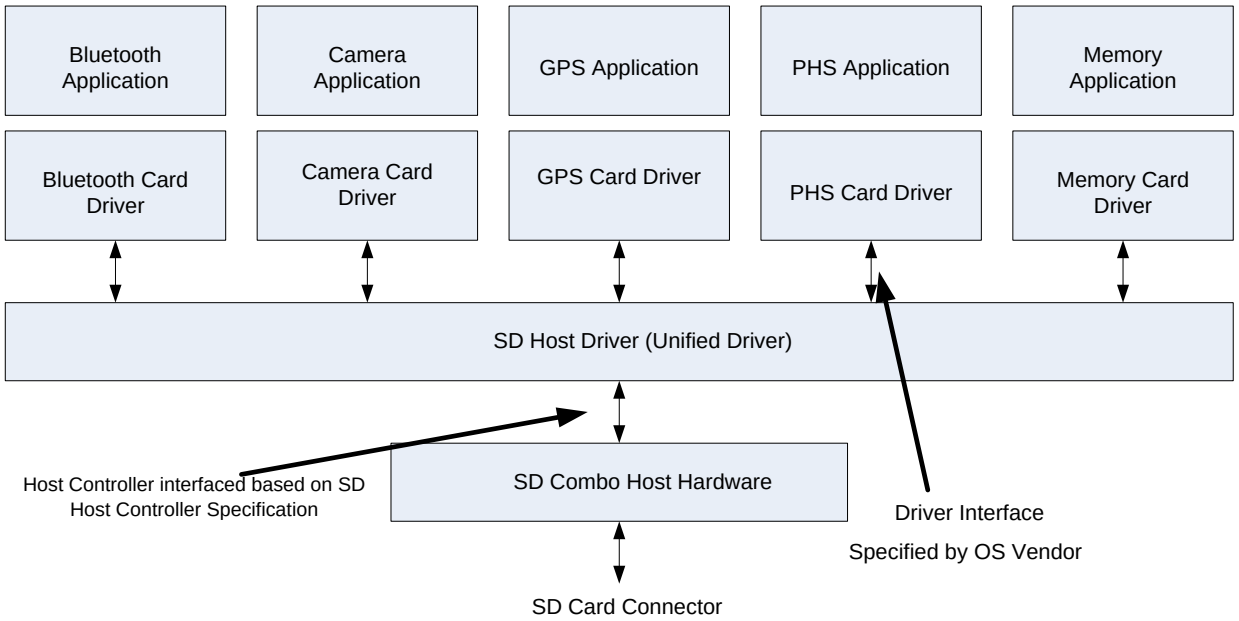
Hardware

The role of the hardware controller is to send out the programmed command, store the response, and make it visible to software. It updates the status registers and generates interrupts when attention is required. Software may also configure it for DMA operation.

Software

The software driver determines which type of card is inserted in the slot by sending the initialization commands and observing the responses received from the card. After identifying the card that is inserted, it should only program the corresponding set of commands that are applicable. It can enable interrupts for which notification is desired.

Figure 114: Host Hardware and Driver Architecture



29.4 Caveats and Assumptions

- A single SD/MMC controller handles only one device at a time.
- The software should configure SDMMC_VENDOR_CLOCK_CNTRL_0_SDMMC_CLK to _DISABLE before turning off the SD/MMC clock (and configure back to _ENABLE after turning on the SD/MMC clock). This is required in order to support asynchronous card interrupts when no clock is supplied to the card.
- PMC can be configured to wake the Tegra K1 device from the LP0 power state based on either:
  - Card Detection pin.

- Asynchronous interrupt on the DAT1 line

- Write protect and card detect logic is implemented in Tegra only for SDMMC1 and SDMMC3.
- Off-card ECC, described in the MMC specification, is not supported.

29.5 SD/MMC Interfaces

Controller	Number of Pinmuxings		Bus Width (Legacy)	Legacy Pads	Voltage (V)	Legacy I/O Clock (MHz)	eMMC4.51	Use Case	Maximum Bandwidth (MBps)
SDMMC1	1	SDMMC1	4	BDSDMEM	3.3/1.8	208		SD/SDIO	104
SDMMC2	2	GMI (SDMMC2A)	8	BDSDMEM	3.3/1.8	208		1st SDIO (with 2 SD), 2nd SDIO (with 1 SD),or 2nd (1.8V) eMMC	104
		PEMMC (SDMMC2B)	8	BDPGLP	3.3/1.8	40 DDR/ 80 SDR		1st SDIO when GMI used for NOR	40
SDMMC3	1	SDMMC3	4	BDSDMEM	3.3/1.8	208		SD/SDIO	104
SDMMC4	1	SDMMC4	8	BDSDMEMLV	1.2/1.8	200		Bootable eMMC	200

29.6 Pinmux Options

Controller	Config	Signal	Ball Name	PINGROUP	Alt
SDMMC1		SDMMC1_CMD	SDMMC1_CMD	sdmmc1_cmd_pm	0
		SDMMC1_CLK	SDMMC1_CLK	sdmmc1_clk_pm	0
		SDMMC1_DAT0	SDMMC1_DAT0	sdmmc1_dat0_pm	0
		SDMMC1_DAT1	SDMMC1_DAT1	sdmmc1_dat1_pm	0
		SDMMC1_DAT2	SDMMC1_DAT2	sdmmc1_dat2_pm	0
		SDMMC1_DAT3	SDMMC1_DAT3	sdmmc1_dat3_pm	0
		SDMMC1_WP_N	SDMMC1_WP_N	sdmmc1_wp_n_pm	0
	Alt1	SDMMC1_CD_N	UART3_CTS_N	uart3_cts_n_pm	1
	Alt2	SDMMC1_CD_N	KB_COL5	kb_col5_pm	2
SDMMC2A		SDMMC2A_CMD	GPIO_PH7	gpio_ph7_pm	0
		SDMMC2A_CLK	GPIO_PK1	gpio_pk1_pm	0
		SDMMC2A_DAT0	GPIO_PH4	gpio_ph4_pm	0
		SDMMC2A_DAT1	GPIO_PI5	gpio_pi5_pm	0
		SDMMC2A_DAT2	GPIO_PH5	gpio_ph5_pm	0
		SDMMC2A_DAT3	GPIO_PH6	gpio_ph6_pm	0
		SDMMC2A_DAT4	GPIO_PK3	gpio_pk3_pm	0
		SDMMC2A_DAT5	GPIO_PK4	gpio_pk4_pm	0
		SDMMC2A_DAT6	GPIO_PI2	gpio_pi2_pm	0
		SDMMC2A_DAT7	GPIO_PI6	gpio_pi6_pm	3
SDMMC2B		SDMMC2B_CMD	CAM_I2C_SDA	cam_i2c_sda	3
		SDMMC2B_CLK	GPIO_PBB3	gpio_pbb3	3
		SDMMC2B_DAT0	GPIO_PBB0	gpio_pbb0	2
		SDMMC2B_DAT1	CAM_I2C_SCL	cam_i2c_scl	3
		SDMMC2B_DAT2	GPIO_PBB5	gpio_pbb5	3
		SDMMC2B_DAT3	CAM_MCLK	cam_mclk	3
		SDMMC2B_DAT4	GPIO_PBB6	gpio_pbb6	3
		SDMMC2B_DAT5	GPIO_PBB7	gpio_pbb7	3
		SDMMC2B_DAT6	GPIO_PCC2	gpio_pcc2	3
		SDMMC2B_DAT7	GPIO_PCC1	gpio_pcc1	3
SDMMC3		SDMMC3_CMD	SDMMC3_CMD	sdmmc3_cmd_pm	0
		SDMMC3_CLK	SDMMC3_CLK	sdmmc3_clk_pm	0
		SDMMC3_DAT0	SDMMC3_DAT0	sdmmc3_dat0_pm	0
		SDMMC3_DAT1	SDMMC3_DAT1	sdmmc3_dat1_pm	0
		SDMMC3_DAT2	SDMMC3_DAT2	sdmmc3_dat2_pm	0
		SDMMC3_DAT3	SDMMC3_DAT3	sdmmc3_dat3_pm	0
	Alt2	SDMMC3_WP_N	KB_COL4	kb_col4_pm	2
	Alt2	SDMMC3_WP_N	GPIO_PCC2	gpio_pcc2_pm	2
		SDMMC3_CD_N	KB_COL5	kb_col5_pm	2

SDMMC4		SDMMC4_CMD	SDMMC4_CMD	sdmmc4_cmd_pm	0
		SDMMC4_CLK	SDMMC4_CLK	sdmmc4_clk_pm	0
		SDMMC4_DAT0	SDMMC4_DAT0	sdmmc4_dat0_pm	0
		SDMMC4_DAT1	SDMMC4_DAT1	sdmmc4_dat1_pm	0
		SDMMC4_DAT2	SDMMC4_DAT2	sdmmc4_dat2_pm	0
		SDMMC4_DAT3	SDMMC4_DAT3	sdmmc4_dat3_pm	0
		SDMMC4_DAT4	SDMMC4_DAT4	sdmmc4_dat4_pm	0
		SDMMC4_DAT5	SDMMC4_DAT5	sdmmc4_dat5_pm	0
		SDMMC4_DAT6	SDMMC4_DAT6	sdmmc4_dat6_pm	0
		SDMMC4_DAT7	SDMMC4_DAT7	sdmmc4_dat7_pm	0

29.6.1 SD CD# and SDWP# Pins

The removable SD cards' Card Detect and Write protect pins apply to SDMMC1 and SDMMC3 only. Tegra K1 devices provide the below pinmux options for these pins.

Controller	Signal	Ball Name	PINGROUP	Alt
SDMMC1	SDMMC1_WP_N	SDMMC1_WP_N	sdmmc1_wp_n_pm	0
	SDMMC1_CD_N	UART3_CTS_N	uart3_cts_n_pm	1
	SDMMC1_CD_N	KB_COL5	kb_col5_pm	2
SDMMC3	SDMMC3_WP_N	KB_COL4	kb_col4_pm	2
	SDMMC3_WP_N	GPIO_PCC2	gpio_pcc5_pm	2
	SDMMC3_CD_N	KB_COL5	kb_col5_pm	2

29.7 Programming Guidelines

This section assumes a SD Host driver complying with the SD Specifications' Part A2, SD Host Control Standard Specification. The following subsections detail the necessary changes required for fully featured SD (and eMMC boot mode) operation.

29.7.1 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:

29.7.1.1 General

These settings apply to all SD/MMC controllers in use:

- Enable tuning in SDR50 by setting SDMMC_VENDOR_CLOCK_CNTRL_0_SDR50_TUNING_OVERRIDE.
- Enable SDR104 (HS200 for eMMC application) support by setting UHS_MODE_SEL to SDR104 in HOST_CONTROL_2 (offset 03Eh) register.
- In DDR50 mode, SD clock divisor should always be 2. That means the host clock should be double that of the card clock for data to be sampled properly.
  - Register SDMMC_SW_RESET_TIMEOUT_CTRL_CLOCK_CONTROL_0_SDCLK_FREQUENCYSELECT value would be 0x1.

- Clamp clocks during resets by setting these two values:
  - SDMMC_VENDOR_CLOCK_CNTRL_0_PADPIPE_CLKEN_OVERRIDE_NORMAL
  - SDMMC_VENDOR_CLOCK_CNTRL_0_SPI_MODE_CLKEN_OVERRIDE_NORMAL

Loopback Clock Selection for SDMMC3 only:

For all modes that require tuning (SDR104, SDR50), disable external loopback on SDMMC3.

For all modes that are non-tunable (DS, HS, SDR12, SDR25, DDR50), enable external loopback on SDMMC3.

The following are the register settings to enable/disable external loopback on SDMMC3 for Tegra K1 devices:

reg SDMMC_VENDOR_MISC_CNTRL_0 SDMMC_SPARE1 0xFFFF// Enables external loopback on SDMMC3
reg SDMMC_VENDOR_MISC_CNTRL_0 SDMMC_SPARE1 0xFFFD// Disables external loopback and uses internal loopback on SDMMC3

29.7.1.2 SDMMC1

These settings are specific to the SDMMC1 controller:

Supply Voltage	33 Ohm		50 Ohm		66 Ohm		100 Ohm	
	UP	DN	UP	DN	UP	DN	UP	DN
1.8 V	6'h46	6'h36	6'h2A	6'h20	6'h22	6'h17	6'h13	6'hB
3.3 V	-		6'h24	6'h14	-		-	

UP - APB_MISC_GP_SDIO1CFGPADCTRL_0_CFG2TMC_SDIO1CFG_CAL_DRVUP

DN - APB_MISC_GP_SDIO1CFGPADCTRL_0_CFG2TMC_SDIO1CFG_CAL_DRVDN

- APB_MISC_GP_SDIO1CFGPADCTRL_0_CFG2TMC_SDIO1CFG_CAL_DRVDN_SLWR **= 2'h0**
- APB_MISC_GP_SDIO1CFGPADCTRL_0_CFG2TMC_SDIO1CFG_CAL_DRVUP_SLWF **= 2'h0**
  SDMMC_SDMEMCOMPPADCTRL_0_SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL **= 4'h7**
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_ENABLE_ENABLED
- The following pull-up and pull-down offsets should be used along with the auto-calibration results for both 3.3V and 1.8V modes. (Note: the offsets are required to eliminate the overshoot/undershoot seen when 33Ω calibration resistors are used. For a 50 Ω driver no offset is required for all modes.)
  - SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = 8'h76
  - SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = **8'h76**

- 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 kΩ pull-up is used on the CMD line):
    - PINMUX_AUX_SDMMC1_CMD_0_PULL_UP_NORMAL
  - If using SD or SDIO:
    - PINMUX_AUX_SDMMC1_CMD_0_PULL_UP_PULL_UP
  - PINMUX_AUX_SDMMC1_DAT3_0_PUPD_PULL_UP
  - PINMUX_AUX_SDMMC1_DAT2_0_PUPD_PULL_UP
  - PINMUX_AUX_SDMMC1_DAT1_0_PUPD_PULL_UP
  - PINMUX_AUX_SDMMC1_DAT0_0_PUPD_PULL_UP

29.7.1.3 SDMMC2A

These settings are specific to the SDMMC2A controller:

Supply Voltage	33 Ohm		50 Ohm		66 Ohm		100 Ohm	
	UP	DN	UP	DN	UP	DN	UP	DN
1.8 V	6'h46	6'h36	6'h2A	6'h20	6'h22	6'h17	6'h13	6'hB
3.3 V	-		6'h24	6'h14	-		-	

UP - APB_MISC_GP_ATCFG6PADCTRL_0_CFG2TMC_ATCFG6_CAL_DRVUP

DN - APB_MISC_GP_ATCFG6PADCTRL_0_CFG2TMC_ATCFG6_CAL_DRVDN

- APB_MISC_GP_ATCFG6PADCTRL_0_CFG2TMC_SDIO2CFG_CAL_DRVDN_SLWR **= 2'h0**
- APB_MISC_GP_ATCFG6PADCTRL_0_CFG2TMC_SDIO2CFG_CAL_DRVUP_SLWF **= 2'h0**
- SDMMC_SDMEMCOMPPADCTRL_0_SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL **= 4'h7**
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_ENABLE_ENABLED
- The following pull-up and pull-down offsets should be used along with the auto-calibration results for both 3.3V and 1.8V modes. (Note: the offsets are required to eliminate the overshoot/undershoot seen when 33Ω calibration resistors are used. For a 50Ω driver no offset is required for all modes.)
  - SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = **8'h76**
  - SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = **8'h76**

- If using the SDMMC2 controller's external interface shared with the GMI pins, correctly enable the integrated pullup/down resistors there:
  - PINMUX_AUX_ GMI_CLK_0 _PUPD_PUUL_DOWN
  - If using eMMC (a stronger external 4.7 kOhm pull-up is used on the CMD line):
    - PINMUX_AUX_GMI_AD15_0_PUPD_NORMAL
  - If using SD or SDIO:
    - PINMUX_AUX_GPIO_PH7_0_PUPD_PULL_UP
  - PINMUX_AUX_GPIO_PH4_0_PUPD_PULL_UP
  - PINMUX_AUX_GPIO_PH5_0_PUPD_PULL_UP
  - PINMUX_AUX_GPIO_PH6_0_PUPD_PULL_DOWN
  - PINMUX_AUX_GPIO_PI2_0_PUPD_PULL_UP
  - PINMUX_AUX_GPIO_PI6_0_PUPD_PULL_UP
  - PINMUX_AUX_GPIO_PI5_0_PUPD_PULL_UP
  - PINMUX_AUX_GPIO_PK3_0_PUPD_PULL_UP
  - PINMUX_AUX_GPIO_PK4_0_PUPD_PULL_UP

29.7.1.4 SDMMC2B

These settings are specific to the SDMMC2B controller:

Supply Voltage	50 Ohm	
	UP	DN
1.8 V	6'd14	6'd8
3.3 V	6'd5	6'd4

CLK, CMD, DAT0, DAT1, and DAT6:

- APB_MISC_GP_GMECFGPADCTRL_0_CFG2TMC_GMECFG_CAL_DRVUP
- APB_MISC_GP_GMECFGPADCTRL_0_CFG2TMC_GMECFG_CAL_DRVDN
- APB_MISC_GP_GMECFGPADCTRL_0_CFG2TMC_GMECFG_CAL_DRVDN_SLWR **= 2'h0**
- APB_MISC_GP_GMECFGPADCTRL_0_CFG2TMC_GME2CFG_CAL_DRVUP_SLWF **= 2'h0**

DAT2, DAT4, and DAT5:

- APB_MISC_GP_GMFCFGPADCTRL_0_CFG2TMC_GMFCFG_CAL_DRVUP
- APB_MISC_GP_GMFCFGPADCTRL_0_CFG2TMC_GMFCFG_CAL_DRVDN
- APB_MISC_GP_GMFCFGPADCTRL_0_CFG2TMC_GMFCFG_CAL_DRVDN_SLWR **= 2'h0**
- APB_MISC_GP_GMFCFGPADCTRL_0_CFG2TMC_GMF2CFG_CAL_DRVUP_SLWF **= 2'h0**

DAT3:

- APB_MISC_GP_GMGCFGPADCTRL_0_CFG2TMC_GMGCFG_CAL_DRVUP
- APB_MISC_GP_GMGCFGPADCTRL_0_CFG2TMC_GMGCFG_CAL_DRVDN
- APB_MISC_GP_GMGCFGPADCTRL_0_CFG2TMC_GMGCFG_CAL_DRVDN_SLWR **= 2'h0**
- APB_MISC_GP_GMGCFGPADCTRL_0_CFG2TMC_GMG2CFG_CAL_DRVUP_SLWF **= 2'h0**

DAT7:

- APB_MISC_GP_GMHCFGPADCTRL_0_CFG2TMC_GMHCFG_CAL_DRVUP
- APB_MISC_GP_GMHCFGPADCTRL_0_CFG2TMC_GMHCFG_CAL_DRVDN
- APB_MISC_GP_GMHCFGPADCTRL_0_CFG2TMC_GMHCFG_CAL_DRVDN_SLWR **= 2'h0**
- APB_MISC_GP_GMHCFGPADCTRL_0_CFG2TMC_GMH2CFG_CAL_DRVUP_SLWF **= 2'h0**
    - SDMMC_SDMEMCOMPPADCTRL_0_SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL = **4'h7**
    - SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_ENABLE_ENABLED
    - SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = **8'h0**
    - SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = **8'h0**

- If using the SDMMC2B controller's external interface shared with the CAM/PEMMC pins, correctly enable the integrated pullup/down resistors there:
    - PINMUX_AUX_ GPIO_PBB3_0 _PUPD_PUUL_DOWN
    - If using eMMC (a stronger external 4.7 kOhm pull-up is used on the CMD line):
        - PINMUX_AUX_CAM_I2C_SDA_0_PUPD_NORMAL
    - If using SD or SDIO:
        - PINMUX_AUX_CAM_I2C_SDA_0_PUPD_PULL_UP
    - PINMUX_AUX_GPIO_PBB0_0_PUPD_PULL_UP
    - PINMUX_AUX_CAM_I2C_SCL_0_PUPD_PULL_UP
    - PINMUX_AUX_GPIO_PBB5_0_PUPD_PULL_UP
    - PINMUX_AUX_CAM_MCLK_0_PUPD_PULL_DOWN
    - PINMUX_AUX_GPIO_PBB6_0_PUPD_PULL_UP
    - PINMUX_AUX_GPIO_PBB7_0_PUPD_PULL_UP
    - PINMUX_AUX_GPIO_PCC2_0_PUPD_PULL_UP
    - PINMUX_AUX_GPIO_PCC1_0_PUPD_PULL_UP

29.7.1.5 SDMMC3

These settings are specific to the SDMMC3 controller:

Supply Voltage	33 Ohm		50 Ohm		66 Ohm		100 Ohm	
	UP	DN	UP	DN	UP	DN	UP	DN
1.8 V	6'h46	6'h36	6'h2A	6'h20	6'h22	6'h17	6'h13	6'hB
3.3 V	-		6'h24	6'h14	-		-	

UP - APB_MISC_GP_SDIO3CFGPADCTRL_0_CFG2TMC_SDIO3CFG_CAL_DRVUP

DN - APB_MISC_GP_SDIO3CFGPADCTRL_0_CFG2TMC_SDIO3CFG_CAL_DRVDN

- APB_MISC_GP_SDIO3CFGPADCTRL_0_CFG2TMC_SDIO3CFG_CAL_DRVDN_SLWR = **2'h0**
- APB_MISC_GP_SDIO3CFGPADCTRL_0_CFG2TMC_SDIO3CFG_CAL_DRVUP_SLWF = **2'h0**
- SDMMC_SDMEMCOMPPADCTRL_2_SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL = **4'h7**
  SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_ENABLE_ENABLED
- The following pull-up and pull-down offsets should be used along with the auto-calibration results for both 3.3V and 1.8V modes. (Note: the offsets are required to eliminate the overshoot/undershoot seen when 33Ω calibration resistors are used. For a 50Ω driver no offset is required for all modes.)
  - SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = **8'h76**
  - SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = **8'h76**

- If using the SDMMC3 interface, correctly enable the integrated pullup/down resistors there:
  - PINMUX_AUX_SDMMC3_CLK_0_PUPD_NORMAL
  - If using eMMC (a stronger external 4.7 kOhm pullup is used on the CMD line):
    - PINMUX_AUX_SDMMC3_CMD_0_PUPD_NORMAL
  - If using SD or SDIO:
    - PINMUX_AUX_SDMMC3_CMD_0_PUPD_PULL_UP
  - PINMUX_AUX_SDMMC3_DAT3_0_PUPD_PULL_UP
  - PINMUX_AUX_SDMMC3_DAT2_0_PUPD_PULL_UP
  - PINMUX_AUX_SDMMC3_DAT1_0_PUPD_PULL_UP
  - PINMUX_AUX_SDMMC3_DAT0_0_PUPD_PULL_UP

29.7.1.6 SDMMC4

These settings are specific to the SDMMC4 controller:

Supply Voltage	33 Ohms		50 Ohms	
	UP	DN	UP	DN
1.8 V	6'h07	6'h07	6'h02	6'h01
1.2 V	6'h0E	6'h0E	6'h05	6'h05

UP - APB_MISC_GP_GMACFGPADCTRL_0_CFG2TMC_GMACFG_CAL_DRVUP

DN - APB_MISC_GP_GMACFGPADCTRL_0_CFG2TMC_GMACFG_CAL_DRVDN

- APB_MISC_GP_GMACFGPADCTRL_0_CFG2TMC_GMACFG_CAL_DRVDN_SLWR **= 2'h0**
- APB_MISC_GP_GMACFGPADCTRL_0_CFG2TMC_GMACFG_CAL_DRVUP_SLWF **= 2'h0**
- SDMMC_SDMEMCOMPPADCTRL_0_SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL **= 4'h7**
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_ENABLE_ENABLED
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = **8'h2**
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = **8'h2**
- If using the SDMMC4 interface, correctly enable the integrated pullup/down resistors there:
- PINMUX_AUX_SDMMC4_CLK_0_PUPD_PULL_DOWN
- If using eMMC (a stronger external 4.7 kOhm pullup is used on the CMD line):
  - PINMUX_AUX_SDMMC4_CMD_0_PUPD_NORMAL
- If using SD or SDIO:
  - PINMUX_AUX_SDMMC4_CMD_0_PUPD_PULL_UP
- PINMUX_AUX_SDMMC4_DAT7_0_PULL_UP_PULL_UP
- PINMUX_AUX_SDMMC4_DAT6_0_PULL_UP_PULL_UP
- PINMUX_AUX_SDMMC4_DAT5_0_PULL_UP_PULL_UP
- PINMUX_AUX_SDMMC4_DAT4_0_PULL_UP_PULL_UP
- PINMUX_AUX_SDMMC4_DAT3_0_PULL_UP_PULL_UP
- PINMUX_AUX_SDMMC4_DAT2_0_PULL_UP_PULL_UP
- PINMUX_AUX_SDMMC4_DAT1_0_PULL_UP_PULL_UP
- PINMUX_AUX_SDMMC4_DAT0_0_PULL_UP_PULL_UP

29.7.2 Tap Values

Software should select the proper tap values for outbound (SDMMC_VENDOR_CLOCK_CNTRL_0_TRIM_VAL) and inbound (SDMMC_VENDOR_CLOCK_CNTRL_0_TAP_VAL) datapaths.

The value of the tap depends on speed modes.

29.7.2.1 Inbound Tap Values (SDMMC_VENDOR_CLOCK_CNTRL_0_TAP_VAL)

The below tap values can be used up to SDR25 speeds.

Controller	IB Tap Val
SDMMC1	0
SDMMC2A	0

Controller	IB Tap Val
SDMMC2B	0
SDMMC3	0
SDMMC4	0

29.7.2.2 DDR52 (eMMC) Tap Settings

Controller	IB Tap Val
SDMMC1	2
SDMMC2A	2
SDMMC2B	1
SDMMC3	0
SDMMC4	4

Note: Software should use tuning procedure to determine the tap value in SDR50 and SDR104/HS200 modes.

29.7.2.3 Outbound Tap Values (SDMMC_VENDOR_CLOCK_CNTRL_0_TRIM_VAL)

The below settings are valid in all speed modes for SDMMC1, SDMMC2A, SDMMC2B and SDMMC3 .

Controller	OB TRIM Val
SDMMC1	2
SDMMC2A	3
SDMMC2B	3
SDMMC3	3

The below settings are valid only for SDMMC4.

Controller	OB TRIM Val	Speed Mode
SDMMC4	4	SDR26, SDR52, HS-200
SDMMC4	0	HS-DDR (DDR52)

29.7.3 eMMC Boot Modes

The section of this document on the boot process describes the boot process in more detail, but note that only the SDMMC4 controller can be used as the eMMC boot device.

29.7.4 Boot Option1

1. The clock to the card needs to be configured to 26 MHz.
2. Program the number of blocks. Configuring block length has no effect; it is always fixed to 512 bytes.
3. Configuring the data transfer direction has no impact, since it taken as CARD to HOST for boot mode.
4. Select either SDMA or PIO mode.
5. Configure SDMMC_VENDOR_BOOT_ACK_TIMEOUT_0 – The maximum timeout value that needs to be programmed is 50ms.The value programmed should exclude the 74 cycles that are required to enter boot mode.
6. Configure SDMMC_VENDOR_BOOT_DAT_TIMEOUT_0 – The max timeout value that needs to be programmed is 1 second .The value programmed should exclude the 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_SDMMC_DBG0_0 [0]. Software shouldn’t program commands until this bit is zero.

29.7.5 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.

29.7.6 Card Detect and Write Protect

Only SDMMC1 and SDMMC3 support removable SD cards. The standard SD Host, SDCD# pin act as card detect pin and CDWP# pin act as Write protect pin. The SD Card State registers can be ignored for embedded on-board devices.

PIN MUX detail for CD and WP pins for SDMMC1 and SDMMC3.

PINMUX_AUX_SDMMC1_WP_N_0_PUPD_PULL_UP

PINMUX_AUX_KB_COL4_0_PUPD_PULL_UP - This is the Pinmux option for SDMMC1 CD

PINMUX_AUX_KB_COL4_0_PUPD_PULL_UP - This is the Pinmux option for SDMMC3 WP

PINMUX_AUX_SDMMC3_CD_N_0_PUPD_PULL_UP

Additionally, an abort command should be issued if a card is removed during an SD transfer.

29.7.7 Improving Non-DMA (PIO) Performance

When transferring more than a single 512 byte block size, PIO performance is reduced because the standard process waits for the buffer ready interrupt before beginning to move the next block of data.

Next block data movement can instead be pipelined on a four-byte word basis. To do this, software should waiting for SDMMC_PRESENT_STATE_0’s BUFFER_READ_EN or BUFFER_WRITE_EN fields to assert instead of the buffer ready interrupt. These status fields signify that the next four-byte word can be read from or written to the internal 512 byte buffer.

29.7.8 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.

Signal Voltage Switching happens after setting 1.8V signal enable in the host control 2 register. Switching the voltage from 3.3V to 1.8V, UHS-I signaling requires switching the external power supply.

After setting 1.8V Signal Enable in the *Host Control 2* register, the SD host driver communicates with the hardware platform to change the SDMMC1, SDMMC2 and SDMMC3 I/O voltage from 3.3 to 1.8V. This is typically via I2C control of an external “PMIC” (Power Management IC).

When resetting the controller (e.g., when an SD3.0 card in 1.8V mode has been removed), the SD host driver again communicates with the system software/platform to increase the SD I/O voltage back to 3.3V.

29.7.8.1 Initialization

Before the standard SD driver begins running:

1. Route the SDMMC controller's system interrupt to the appropriate driver.
2. Write 1 to SDMMC_VENDOR_SYS_SW_CNTRL_0_VOLT_SWITCH_INT_STATUS to clear any latent interrupts.
3. Set SDMMC_VENDOR_SYS_SW_CNTRL_0_VOLT_SWITCH_INT _ENABLE to enable the system interrupt on an SD3.0 UHS-I Voltage Switch event.

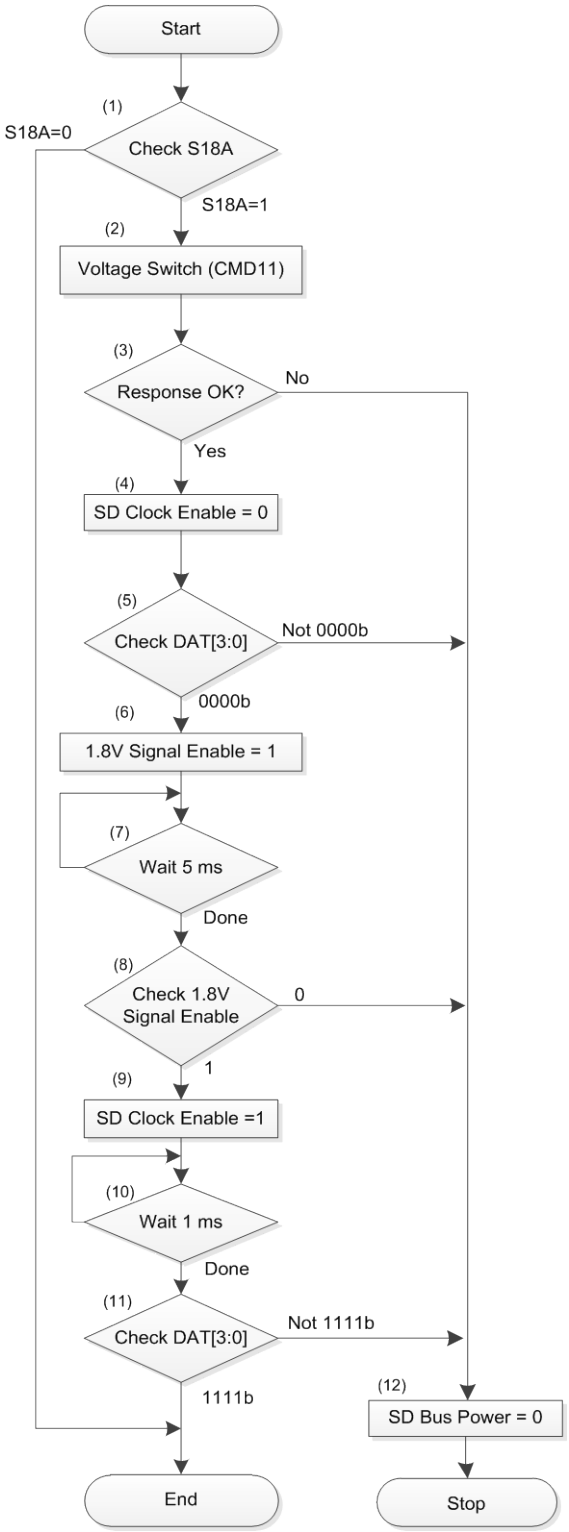
29.7.8.2 Entry

Once an SDMMC controller generates a system interrupt:

- Confirm the SDMMC System interrupt was from a voltage switch event by reading SDMMC_VENDOR_SYS_SW_CNTRL_0_VOLT_SWITCH_INT _STATUS.

29.7.8.3 Voltage Switch Procedure

The following figure shows the standard voltage switch procedure.



29.7.8.1 Exit

Write 1 to SDMMC_VENDOR_SYS_SW_CNTRL_0_VOLT_SWITCH_INT _STATUS to clear the interrupt condition.

29.7.9 Tuning

Tuning is required because SD/MMC controllers do not automatically vary the RX clock trim each time that Execute Tuning is set to 1; the SD Host driver must do this manually.

SD 3.0 cards can operate at frequencies up to 208 MHz. For the card to operate at high frequencies, the correct tap value must be programmed. The tap value in the vendor clock control register controls the track delay so that the input data is sampled correctly. The appropriate tap delay for the given platform is found using frequency tuning. The passing tap value range can be found by incrementing the tap value after each iteration and running frequency tuning.

In frequency tuning, CMD19 for SD and CMD21 for eMMC (emmc4.51) card is issued to the card and wait for the Buffer Read Ready interrupt. Once the interrupt is generated, in the HOST_CONTROL_2 register (offset 03Ch), check whether the EXECUTE_TUNING bit is cleared and the SAMPLING_CLK_SELECT bit is set. If both 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.

29.7.9.1 Initialization

Before the standard SD drive begins running:

1. Route the SD/MMC controller's system interrupt to the appropriate driver.
2. Write 1 to SDMMC_VENDOR_SYS_SW_CNTRL_0_TUNING_SYS_INT_STATUS to clear any latent interrupts.
3. Set SDMMC_VENDOR_SYS_SW_CNTRL_0_TUNING_SYS_INT_ENABLE to enable the system interrupt on an SD3.0 UHS-I tuning start event.

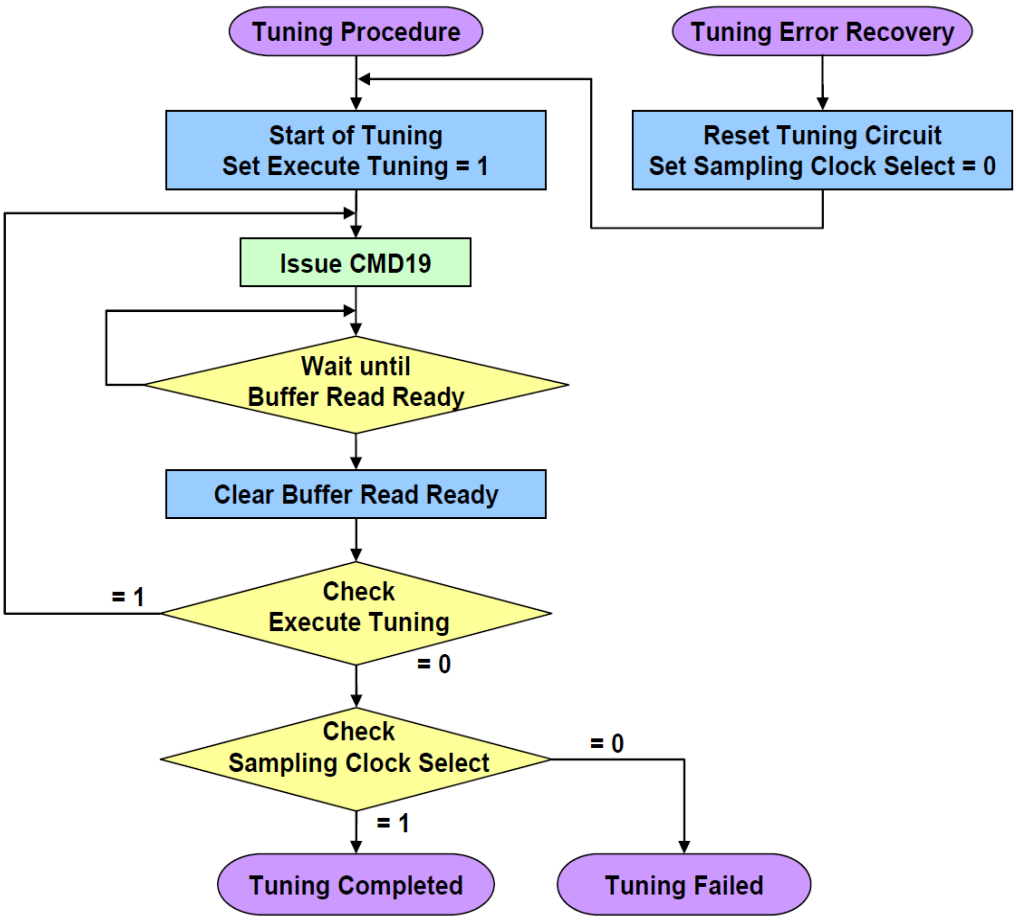
29.7.9.2 Entry

Once an SD/MMC system interrupt is received:

1. Confirm the SDMMC System interrupt was from a tuning event by reading SDMMC_VENDOR_SYS_SW__CNTRL_0_TUNING_SYS_INT_STATUS.
2. Disable the future tuning interrupts (to avoid interrupting yourself) by clearing SDMMC_VENDOR_SYS_SW_CNTRL_0_TUNING_SYS_INT_ENABLE.
3. Write 1 to SDMMC_VENDOR_SYS_SW_CNTRL_0_TUNING_SYS_INT_STATUS to clear the interrupt condition.
4. Disable the Buffer Read Ready interrupt to keep the standard SD Host Driver inactive until the tuning sequence is complete: Clear SDMMC_INTERRUPT_STATUS_ENABLE_0_ BUFFER_READ_READY.

29.7.9.3 Tuning Procedure

The figure below shows the standard tuning procedure.



29.7.9.4 Exit

1. Write 1 to SDMMC_VENDOR_SYS_SW_CNTRL_0_TUNING_SYS_INT_STATUS to clear any pending tuning interrupts.
2. Set SDMMC_VENDOR_SYS_SW_CNTRL_0_TUNING_SYS_INT_ENABLE to enable the system interrupt on any future SD3.0 UHS-I tuning start events.
3. Signal to the SD Host driver that tuning has completed by asserting SDMMC_VENDOR_SYS_SW_CNTRL_0_ASSERT_BUFF_RD_RDY_INT.

29.7.10 SD Bus Power

When the standard SD driver changes the Power Control Register's SD Bus Power field, the hardware needs assistance from platform software to implement the change via the PMIC.

29.7.10.1 Initialization

Before the standard SD drive begins running:

1. Route the SD/MMC controller's system interrupt to the appropriate driver.
2. Write 1 to SDMMC_VENDOR_SYS_SW_CNTRL_0_SD_BUS_POWER_ON_OFF_INT_STATUS to clear any latent interrupts.

3. Set SDMMC_VENDOR_SYS_SW_CNTRL_0_SD_BUS_POWER_ON_OFF_INT_ENABLE to enable the system interrupt on an SD3.0 UHS-I Bus Power change event.

29.7.10.2 Entry

Once an SD/MMC system interrupt is received:

1. Confirm the SDMMC System interrupt was from a Bus Power change event by reading SDMMC_VENDOR_SYS_SW_CNTRL_0_SD_BUS_POWER_ON_OFF_INT_STATUS.
2. Write 1 to SDMMC_VENDOR_SYS_SW_CNTRL_0_SD_BUS_POWER_ON_OFF_INT_STATUS to clear the interrupt condition.
3. Read SDMMC_POWER_CONTROL_HOST_0_SD_BUS_POWER to determine the power state change request.

   If Bus Power is cleared, power off the SD power supply via the PMIC. Otherwise, if Bus Power is set, power on the SD power supply via the PMIC.

29.7.10.3 Exit

No actions required.

29.8 SD/MMC Registers

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

29.8.1 Standard Registers

Note: For standard registers’ usage, refer to the SD Host Controller Specification version 4.00.

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:

29.8.1.1 Present State Register

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

Bit	Reset	Description
28:25	X	DAT_7_4_LINE_LEVEL: This status reflects the DAT[7:4] lines’ state for debug.

29.8.1.2 Interrupt Status Register

Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000xxxxxxxx00000000)

Bit	Reset	Description
31:30	0	VEND_SPEC_ERR Bit 1: BOOT_ACK_ERR: Occurs when Boot Ack Status is not equal to ‘010’ Bit 0: BOOT_ACK_TIMEOUT_ERR: Occurs when Boot Ack is not received within the programmed number of cycles.

29.8.1.3 Interrupt Status Enable Register

Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000xxx0xxx000000000)

Bit	Reset	Description
31:30	0	VENDOR_SPECIFIC_ERR 0 = Enable neither BOOT_ACK_ERR nor BOOT_ACK_TIMEOUT_ERR statuses. 1 = Enable only BOOT_ACK_TIMEOUT_ERR status. 2 = Enable only BOOT_ACK_ERR status. 3 = Enable both BOOT_ACK_ERR and BOOT_ACK_TIMEOUT_ERR statuses.

29.8.1.4 Interrupt Signal Enable Register

Offset: 0x38 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000xxx0xxx000000000)

Bit	Reset	Description
31:30	0	VENDOR_SPECIFIC_ERR 0 = Enable neither BOOT_ACK_ERR nor BOOT_ACK_TIMEOUT_ERR interrupts to CPU. 1 = Enable only BOOT_ACK_TIMEOUT_ERR interrupt to CPU. 2 = Enable only BOOT_ACK_ERR interrupt to CPU. 3 = Enable both BOOT_ACK_ERR and BOOT_ACK_TIMEOUT_ERR interrupts to CPU.

29.8.1.5 Force Event Register

Offset: 0x50 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000xxxxxxxx0x000000)

Bit	Reset	Description
31:30	0	VENDOR_SPECIFIC_ERR_STATUS 0 = No effect 1 = Force a BOOT_ACK_TIMEOUT_ERR event 2 = Force a BOOT_ACK_ERR event. 3 = Force both BOOT_ACK_ERR and BOOT_ACK_TIMEOUT_ERR events.

29.8.2 Vendor Registers

29.8.2.1 SDMMC_VENDOR_CLOCK_CNTRL_0

The following registers are Vendor Specific Registers and are mapped to Vendor Specific Address Space (0x100 - 0x1FF)

Vendor Clock Control Register

TAP_VAL - Tap Value for the input data path trimmer

This determines the tap value needed to sample the input data correctly. The delay per each tap can range from 70ps to 500ps. The following values are recommended on different SD/MMC interfaces for all modes up to SDR25:

- SDMMC1 - 2
- SDMMC2 - 2
- SDMMC3 - 3
- SDMMC4 - 5

For SDR50 and above modes, the tap value is determined by the tuning procedure.

BASE_CLK_FREQ: Software should program the actual clock frequency programmed in the CAR registers CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC*_0 for each SD/MMC controller. This should be done after every time

SDMMC is reset and after every soft reset. This is important because all SD/MMC controllers follow the same register map, but could be programmed with different frequencies depending on the use case.

SPI_MODE_CLKEN_OVERRIDE: Software should always program this to 0 (NORMAL).

PADPIPE_CLKEN_OVERRIDE: Software should always program this to 0 (NORMAL).

Offset: 0x100 │ Read/Write: R/W │ Reset: 0x0005d00d (0bxxx00000000010111010000x0001101)

Bit	Reset	SW Default	Description
28:24	0x0	NONE	TRIM_VAL: Reserved. Software should not change these bits.
23:16	0x5	NONE	TAP_VAL: Tap value for input data path trimmer.
15:8	0xd0	NONE	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	0x1	SDR50_TUNING_OVERRIDE: override the SDR50_TUNING capabilities bit. Software should only set this bit if it is required to use Tuning for SDR50. (only supported for SDMMC1 and SDMMC3) 0 = NORMAL : 0 -> No tuning support advertised for SDR50 mode. 1 = OVERRIDE: 1 -> Tuning support is enabled for SDR50 mode.
3	0x1	0x0	PADPIPE_CLKEN_OVERRIDE: Override for pad macro and pipe macro clken. 0 -> CLKEN is to pad macro and pipe macro can be deasserted along with internal CLKEN. 0 = NORMAL: 0 -> CLKEN is to pad macro and pipe macro can be deasserted along with internal CLKEN. 1 = OVERRIDE: 1 -> CLKEN is kept asserted to pad macro and pipe macro when internal CLKEN is deasserted.
2	0x1	NONE	SPI_MODE_CLKEN_OVERRIDE: Override for CLKEN during SPI_MODE during sw_reset. 0 = NORMAL: 0 -> CLKEN is deasserted while doing sw_reset. 1 = OVERRIDE: 1 -> CLKEN is kept asserted while doing sw_reset.
1	0x0	NONE	INPUT_IO_CLK: Feedback clock is selected by default. Software should not change this. Disabling the Feedback clock will select the Internal Clock that requires different tap value programming. 0 = FEEDBACK 1 = INTERNAL
0	0x1	NONE	SDMMC_CLK: This is set when sdmmc_clk is supplied by the CAR module. Prior to sdmmc_clk switch OFF. This bit should be written '0'. By writing zero, the asynchronous card interrupt is routed to the Interrupt controller. 0 = DISABLE 1 = ENABLE

29.8.2.2 SDMMC_VENDOR_SYS_SW_CNTRL_0

Vendor System SW Control Register

Offset: 0x104 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx000xxxxx0000x00)

Bit	Reset	Description
14	0x0	SD_BUS_POWER_ON_OFF_INT_STATUS: SD_BUS_POWER was changed. System software can use this interrupt to implement power switch. 0 = NO_INT 1 = GEN_INT
13	0x0	VOLT_SWITCH_INT_STATUS: VOLT_18_EN was changed. System software can use this interrupt to implement a UHS-I voltage switch procedure for a standard SD Host driver 0 = NO_INT 1 = GEN_INT
12	0x0	TUNING_SYS_INT_STATUS: CMD19 was issued while EXECUTE_TUNING was set. System software can use this interrupt to implement a UHS-I tuning procedure for a standard SD Host driver. 0 = NO_INT 1 = GEN_INT
6	0x0	SD_BUS_POWER_ON_OFF_INT_ENABLE: Enables a separate system interrupt (i.e., not the standard SDHCI interrupt) when SD_BUS_POWER is changed.

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
5	0x0	VOLT_SWITCH_INT_ENABLE: Enables a separate system interrupt (i.e., not the standard SDHCI interrupt) when VOLT_18_EN is changed. 0 = DISABLE 1 = ENABLE
4	0x0	TUNING_SYS_INT_ENABLE: Enables a separate system interrupt (i.e., not the standard SDHCI interrupt) when CMD19 is issued while EXECUTE_TUNING is set. 0 = DISABLE 1 = ENABLE
3	0x0	ASSERT_BUFF_RD_RDY_INT: Write a 1 to this field to assert sdmmc_interrupt_status_0_buffer_read_ready. Used by the system software that implements the tuning procedure to signal to the standard SD driver that the tuning process has completed 0 = DISABLE 1 = ENABLE
1	0x0	INT_MASK_WHILE_TUNING: to avoid breaking software compatibility, interrupt generation behavior is changed when a this bit is set 0 = DISABLE 1 = ENABLE
0	0x0	SPI_MODE: This is a mirror bit. The SPI mode is set if this bit is set or CMD_XFER_MODE[7] is set. Writing 1 will drive the CS Low and writing zero will deassert the CS signal 0 = DISABLE : 0 -> SPI mode is disabled. 1 = ENABLE: 1 -> SPI mode is enabled.

29.8.2.3 SDMMC_VENDOR_CAP_OVERRIDES_0

Capabilities Override Bits

Offset: 0x10c │ Read/Write: R/W │ Reset: 0x00000007 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx111)

Bit	Reset	Description
2	0x1	VOLTAGE_3_3_V_SUPPORT_OVERRIDE: Voltage support 3_3_V override
1	0x1	VOLTAGE_3_0_V_SUPPORT_OVERRIDE: Voltage support 3_0_V override
0	0x1	VOLTAGE_1_8_V_SUPPORT_OVERRIDE: Voltage support 1_8_V override

29.8.2.4 SDMMC_VENDOR_BOOT_CNTRL_0

Vendor Boot Control Register

This register is used to configure Boot Mode to support MMC v4.3 cards.

Offset: 0x110 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	BOOT_ACK: This bit is used to support Boot Option in MMC 4.3 version cards. If set Boot acknowledgment is given by card else not given by card 0 = DISABLE 1 = ENABLE
0	0x0	BOOT: This bit enables/disable BootOption1.If set BootOption1 is enabled, hardware auto clears it when boot data is done. Writing 0 terminates the BootOption1 0 = DISABLE 1 = ENABLE

29.8.2.5 SDMMC_VENDOR_BOOT_ACK_TIMEOUT_0

Vendor Boot Acknowledgment Timeout Register

Offset: 0x114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	VALUE: If Boot Acknowledgment is not received within the programmed number of cycles, aBoot Acknowledgement Timeout error occurs (VENDOR_SPECIFIC_ERR[0])

29.8.2.6 SDMMC_VENDOR_BOOT_DAT_TIMEOUT_0

Boot Data Timeout Register

Offset: 0x118 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
24:0	0x0	VALUE: If Boot Data is not received within the programmed number of cycles, a Data Timeout error occurs.

29.8.2.7 SDMMC_VENDOR_DEBOUNCE_COUNT_0

Debounce Counter Value Register

The Debounce Counter runs on a 32 KHz clock. Keeping the default value to 100ms = (100 * 32 cycles/1ms) = 3200 cycles for 100ms = 0xC80

Offset: 0x11c | Read/Write: R/W | Reset: 0x00000c80 (0b00000000000000110010000000)

Bit	Reset	Description
23:0	0xc80	VALUE: The number of 32KHz clock cycles is programmed to meet the debounce period of the card slot.

29.8.2.8 SDMMC_VENDOR_MISC_CNTRL_0

Miscellaneous Vendor Control Register

- SDMMC_SPARE0: Spare register bits with reset value of 0
  - SDMMC_SPARE0[0]: SW_RESET_CLKEN_OVERRIDE, override the ‘sdmmc_clken’ when doing SW_RESET if set to 1.
  - SDMMC_SPARE0[1]: When set, allows SD clock to be stopped in the middle of a read data block while in SDR104 mode. Unsafe for at least some SD cards, but may improve SDR104 DMA read performance in some cases.
  - SDMMC_SPARE0[2]: When set, SDR104 support is advertised in SDMMC_CAPABILITIES_HIGHER_0_SDR104
  - SDMMC_SPARE0[3]: When set, SDR50 support is advertised in SDMMC_CAPABILITIES_HIGHER_0_SDR50
  - SDMMC_SPARE0[5]: When 0, masks the pad macro's "high speed" enable to 0, causing the pad macro to always launch data on the falling edge of the clock. This prevents the SD Host driver's setting of SDMMC_POWER_CONTROL_HOST_x_HIGH_SPEED_EN from undesirably affecting the output timing.
  - SDMMC_SPARE0[7:6]: Number of pipe stages.
  - SDMMC_SPARE0[8]: When set, DDR50 support is advertised in SDMMC_CAPABILITIES_HIGHER_0_DDR50
  - SDMMC_SPARE1: Spare register bits with reset value of 1
  - SDMMC_SPARE1[0]: CMD_CONFLICT_DISABLE, disable command line conflict if set to 1.

Offset: 0x120 | Read/Write: R/W | Reset: 0xffff0098 (0b11111111111111110000000010011000)

Bit	Reset	Description
31:16	0xffff	SDMMC_SPARE1: Spare register bits with reset value of 1
15:1	0x4c	SDMMC_SPARE0: Spare register bits with reset value of 0x40
0	0x0	ERASE_TIMEOUT_LIMIT: Erase timeout value. 0 = FINITE: Finite. It is limited to the programmed value in the DATA_TIMEOUT_VALUE 1 = INFINITE: Infinite. Controller would be monitoring until the card is busy.

29.8.2.9 SDMMC_MAX_CURRENT_OVERRIDE_0

Offset: 0x124 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:16	0x0	OVERRIDE_FOR_1_8V: Maximum override for 1.8V VDD1.
15:8	0x0	OVERRIDE_FOR_3_0V: Maximum override for 3.0V VDD1.
7:0	0x0	OVERRIDE_FOR_3_3V: Maximum override for 3.3V VDD1.

29.8.2.10 SDMMC_MAX_CURRENT_OVERRIDE_HI_0

Offset: 0x128 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	OVERRIDE_FOR_1_8V_VDD2: Maximum override for 1.8V VDD2.

29.8.2.11 SDMMC_VENDOR_CLK_GATE_HYSTERESIS_COUNT_0

Vendor Clock Gating Hysteresis Counter Initial Value

Offset: 0x1d0 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx001111)

Bit	Reset	Description
5:0	0xf	CLK_COUNT: Before gating second-level clocks, the controller waits for these many cycles. Recovery is possible if any idle windows are missed in the clken equation

29.8.2.12 SDMMC_VENDOR_PRESET_VAL0_0

Vendor Preset Value Registers

The SD host specification defines one preset value register for each bus speed mode which should be set by the host by some unique method. Preset values vary based on the base frequency used which is under software (Tegra system driver) control. The Tegra System driver configures the BASE_CLK_FREQ in VENDOR_CLOCK_CNTRL register before handling over control to the SD host standard driver.

In a similar way, the system driver should set the below vendor preset values based on the base clock frequency and the desired card clock frequency in each bus speed mode. This should be done after the SDMMC is reset and after every soft reset. This is important because all SDMMC controllers follow the same register map, but can be programmed with different frequencies depending on the use case.

The default values are set assuming base clock frequency = 208 MHz.

Offset: 0x1d4 | Read/Write: R/W | Reset: 0x00201000 (0bxx00000001000000001000000000000)

Bit	Reset	Description
29:20	0x2	SDCLK_FREQ_SEL_HIGH_SPEED: System software programs the 10-bit divider value to generate an SD clock in default speed mode (< 50 MHz,3.3V signaling). This value is readable in the standard via PRESET_SDR12_AND_HIGH_0_SDCLK_FREQ_VAL_LOW register field. The default value is 0x2 assuming a 208 MHz base clock.
19:10	0x4	SDCLK_FREQ_SEL_DEFAULT: System software programs the 10-bit divider value to generate an SD clock in default speed mode (<25 MHz,3.3V signaling). This value is readable in the standard via PRESET_DEFAULT_AND_INIT_0_SDCLK_FREQ_VAL_HIGH register field. The default value is 0x4 assuming a 208 MHz base clock.
9:0	0x0	SDCLK_FREQ_SEL_INIT: System software programs the 10-bit divider value to generate the desired SD clock frequency during initialization. This value is readable in the standard via PRESET_DEFAULT_AND_INIT_0_SDCLK_FREQ_VAL_LOW register field. For example, if a 400KHz SDCLK is desired @base clk freq=48 MHz, this register should be programmed with 0x3C.

29.8.2.13 SDMMC_VENDOR_PRESET_VAL1_0

Offset: 0x1d8 | Read/Write: R/W | Reset: 0x00100804 (0bxx00000000100000000100000000100)

Bit	Reset	Description
29:20	0x1	SDCLK_FREQ_SEL_SDR50: System software programs the 10-bit divider value to generate the SD clock in SDR50 mode (<100 MHz,1.8V signaling). This value is readable in the standard via PRESET_SDR50_AND_SDR25_0_SDCLK_FREQ_VAL_HIGH register field. The default value is 0x1 (gives 2N divider) assuming a 208 MHz base clock.
19:10	0x2	SDCLK_FREQ_SEL_SDR25: System software programs the 10-bit divider value to generate the SD clock in SDR25 mode (<50 MHz,1.8V signaling). This value is readable in the standard via PRESET_SDR50_AND_SDR25_0_SDCLK_FREQ_VAL_LOW register field. The default value is 0x2 assuming a 208 MHz base clock.
9:0	0x4	SDCLK_FREQ_SEL_SDR12: System software programs the 10-bit divider value to generate the SD clockk in SDR12 mode (<25 MHz,1.8V signaling). This value is readable in the standard via PRESET_SDR12_AND_HIGH_0_SDCLK_FREQ_VAL_HIGH register field.

29.8.2.14 SDMMC_VENDOR_PRESET_VAL2_0

Offset: 0x1dc | Read/Write: R/W | Reset: 0x00000800 (0bxxxxxxxxxxxxx00000000100000000000)

Bit	Reset	Description
19:10	0x2	SDCLK_FREQ_SEL_DDR50: System software programs the 10-bit divider value to generate the SD clock in DDR50 mode (<50 MHz,1.8V signaling). This value is readable in the standard via PRESET_DDR50_AND_SDR104_0_SDCLK_FREQ_VAL_HIGH register field. The default value is 0x2 assuming 208 MHz base clock.
9:0	0x0	SDCLK_FREQ_SEL_SDR104: System software programs the 10-bit divider value to generate the SD clock in SDR104 mode (<208 MHz,1.8V signaling). This value is readable in the standard via PRESET_DDR50_AND_SDR104_0_SDCLK_FREQ_VAL_LOW register field.

29.8.2.15 SDMMC_SDMEMCOMPPADCTRL_0

SDMEMCOMP Pad Control Register

If AUTO_CAL_ENABLE is disabled (0), the values in this register are used to drive the DRVUP/DRVDN controls to the SDMMC comp pads.

Offset: 0x1e0 │ Read/Write: R/W │ Reset: 0xa9318007 (0b101010010011x00110000000000000111)

Bit	Reset	SW Default	Description
31	0x1	NONE	PAD_E_INPUT_OR_E_PWRD: Used to control E_INPUT (for SDMMC1/2/3) and E_PWRD (for SDMMC4) input of PU/PD comp pad should be cleared once auto-calibration is done (for power saving). NOTE: E_PWRD = !PAD_E_INPUT_OR_E_PWRD and E_INPUT = PAD_E_INPUT_OR_E_PWRD
30:29	0x1	NONE	PU_PAD_DRV_TYPE
28:27	0x1	NONE	PD_PAD_DRV_TYPE
26:20	0x13	0x22	CFG_SDMEMCOMP_DRVUP: used if AUTOCAL is disabled
18:12	0x18	0x14	CFG_SDMEMCOMP_DRVDN: used if AUTOCAL is disabled
11	0x0	NONE	PU_PAD_E_TEST_OUT
10	0x0	NONE	PD_PAD_E_TEST_OUT
9:7	0x0	NONE	PU_PAD_TEST_SEL
6:4	0x0	NONE	PD_PAD_TEST_SEL
3:0	0x7	NONE	SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL

29.8.2.16 SDMMC_AUTO_CAL_CONFIG_0

SDMEMCOMP pad auto-calibration settings - Valid for SDMMC1/SDMMC3 Instances

AUTO_CAL_SLW_OVERRIDE

- 0 (Normal operation) pad DRVDN/UP_SLWR/F tied to AUTO_CAL output DRDVDN/UP_SLWR/F[1:0] = AUTO_CAL_PULLDOWN/UP[4:3]
- 1 (override) use CFG2TMC_SDIO[1|3]*_DRVDN/UP_SLWR/F pins to control pad slew inputs

AUTO_CAL_OVERRIDE

- 0 (normal operation): use AUTO_CAL_PU/PD_OFFSET as an offset to the calibration state machine setting
- 1 (override): use AUTO_CAL_PU/PD_OFFSET register values directly

Offset: 0x1e4 │ Read/Write: R/W │ Reset: 0x00010000 (0b0000xxxxxxxxx001x0000000x0000000)

Bit	Reset	Description
31	0x0 –	AUTO_CAL_START: Writing a one to this bit starts the calibration state machine. This bit must be set even if the override is set in order to latch in the override value.
30	0x0	AUTO_CAL_OVERRIDE: AUTOCAL override. 0 = NORMAL : 0 (normal operation): use AUTO_CAL_PU/PD_OFFSET as an offset to the calibration state machine setting 1 = OVERRIDE : 1 (override) : use AUTO_CAL_PU/PD_OFFSET register values directly
29	DISABLED	AUTO_CAL_ENABLE: AUTOCAL enable. 0 = DISABLED : 0 (disabled): use sdmmc2tmc_cfg* register settings for pull-up/dn 1 = ENABLED : 1 (normal operation): use SDMMC generated pull-up/dn (override or AUTOCAL)
28	0x0	AUTO_CAL_SLW_OVERRIDE: AUTOCAL slew rate override. 0 = NORMAL : 0 (Normal operation) pad DRVDN/UP_SLWR/F tied to AUTO_CAL output DRDVDN/UP_SLWR/F[1:0] = AUTO_CAL_PULLDOWN/UP[4:3] 1 = OVERRIDE : 1 (override) use CFG2TMC_SDIO[1|3]*_DRVDN/UP_SLWR/F pins to control pad slew inputs
18:16	0x1	AUTO_CAL_STEP: calibration step interval (in microseconds)
14:8	0x0	AUTO_CAL_PD_OFFSET: 2's complement offset for pull-down value
6:0	0x0	AUTO_CAL_PU_OFFSET: 2's complement offset for pull-up value

29.8.2.17 SDMMC_AUTO_CAL_INTERVAL_0

SDMEMCOMP Pad Calibration Interval

Offset: 0x1e8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	AUTO_CAL_INTERVAL: 0: Do calibration once. Otherwise, auto-calibration occurs at intervals equivalent to the programmed number of microseconds.

29.8.2.18 SDMMC_AUTO_CAL_STATUS_0

SDMEMCOMP Pad Calibration Status

Offset: 0x1ec | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	AUTO_CAL_ACTIVE: One when auto calibrate is active - valid only after auto calibrate sequence has completed (AUTO_CAL_ACTIVE == 0)
30:24	X	AUTO_CAL_PULLDOWN_ADJ: Pull-down code sent to pads
22:16	X	AUTO_CAL_PULLUP_ADJ: Pull-up code sent to pads
14:8	X	AUTO_CAL_PULLDOWN: Pull-down code generated by auto-calibration
6:0	X	AUTO_CAL_PULLUP: Pull-up code generated by auto-calibration

29.8.2.19 SDMMC_SDMMC_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control (where applicable) and Clock Gating Control Register

Note: The FIFO timing aspects of this register are no longer supported, but retained for software compatibility.

The clock override/ovr_mode fields of this register control the second-level clock gating for the client and MC side of the MCCIF. All clock gating is enabled by default.

A '1' written to the rclk/wclk override field will result in one of the following:

- With wclk/rclk override mode = LEGACY, the clock reverts to the legacy mode of operation where the clock is on whenever the client clock is enabled.
- With wclk/rclk override mode = ON, the clock is always on inside the MCCIF and PC.

A '1' written to the cclk override field keeps the client's clock always on inside the MCCIF.

Offset: 0x1f4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000xxxxxxxxxxxxxxx)

Bit	Reset	Description
20	0x0	SDMMC_RCLK_OVR_MODE
19	0x0	SDMMC_WCLK_OVR_MODE
18	0x0	SDMMC_CCLK_OVERRIDE
17	0x0	SDMMC_RCLK_OVERRIDE
16	0x0	SDMMC_WCLK_OVERRIDE

29.8.2.20 SDMMC_TIMEOUT_WCOAL_SDMMC_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: 0x1f8 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00110010)

Bit	Reset	Description
7:0	0x32	SDMMCW_WCOAL_TMVAL

[THIS PAGE INTENTIONALLY LEFT BLANK]

30.0 SNOR (GMI) CONTROLLER

The GMI Controller, also known as the Sync-NOR (SNOR) Controller, supports a number of NOR flash memory types, including synchronous NOR and asynchronous NOR. It enables:

- Data transfer between internal and external memory.
- Interface with Synchronous/Asynchronous Flash Memories.

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 GMI/SNOR controller supports DMA transfer between Flash Memory and System Memory.

The GMI/SNOR controller can operate in 2 modes:

- Programmed I/O (PIO) mode – In this case the CPU 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.
- 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.

30.1 Functional Description

The GMI/SNOR Controller supports the following operation:

- Synchronous and Asynchronous Flash Memories 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 Functional Waveforms

Table 129 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® K1 and SNOR 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 115 Synchronous Read (GMI Non-Mux Device)

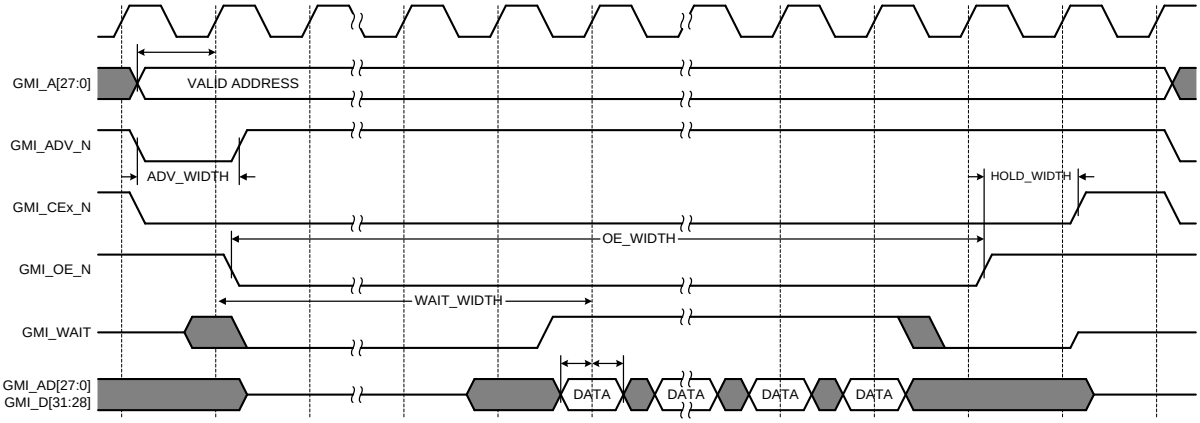


Figure 116 Synchronous Read (GMI Mux-AD 16-bit Data Device)

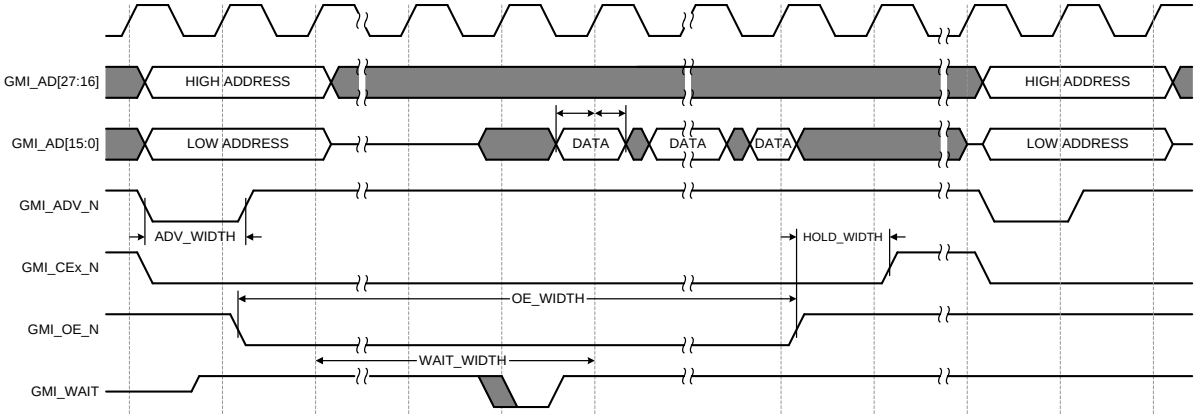


Figure 117 Asynchronous Read (GMI MUX-AD 16-bit Data Device)

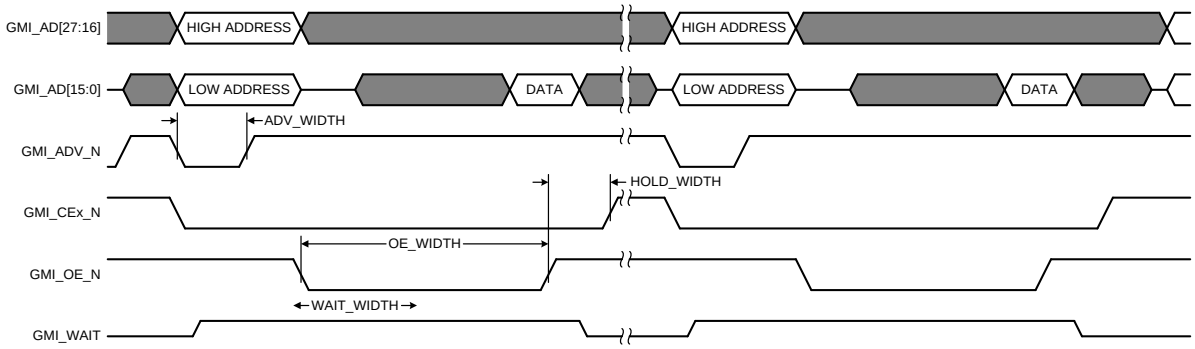
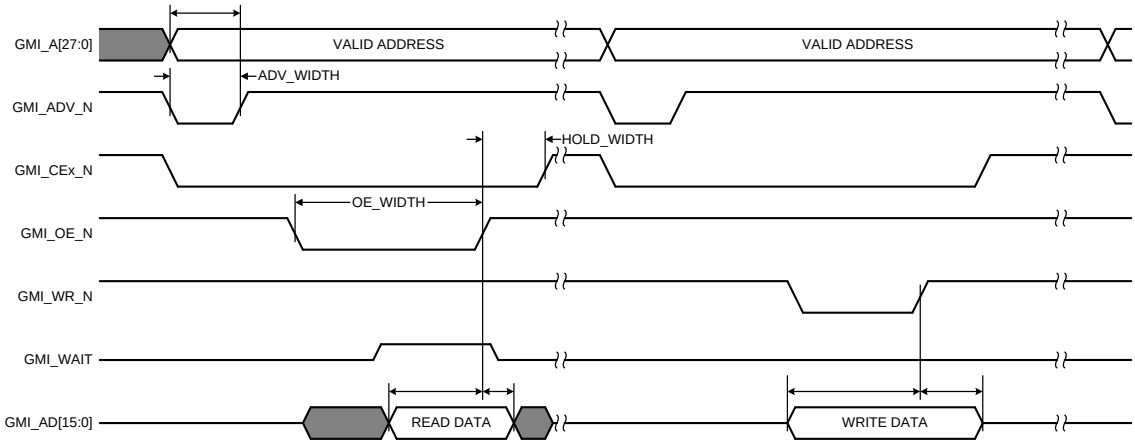


Figure 118 Asynchronous Read Followed by Asynchronous Write (GMI Non-Mux 16-bit Data Device)



30.1.2 GMI/SNOR Memory Mapping

Table 130 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 131 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/SNOR Registers

30.3.1 SNOR_CONFIG_0

SNOR Configuration Register

Offset: 0x0 | 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 Data Bus width Configuration Bit. 0 = NOR16BIT 1 = NOR32BIT
29	0x0	NOR_DEVICE_TYPE: External NOR Memory Type. 0 = SNOR
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. Note: Programming this field to 0 is not used. 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: 0x4 | Read/Write: R/W | Reset: 0xX0X0XXXX (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	Reserved
26	RW	0x0	Reserved
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: 0x8 | 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: 0xc | 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: 0x10 | 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. This field must be programmed with a minimum value of 3.
23:20	0x1	PAGE_SEQ_WIDTH: Page Sequential width indicates the delay cycle between the intra page Read access. This field must be programmed with a minimum value of 2.
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: 0x14 | Read/Write: R/W | Reset: 0x10010103 (0b000100x00000000100000001000000011)

Bit	Reset	Description
31:26	0x4	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. This field needs to be programmed to 0x4 when the RDY_ACTIVE bit in the SNOR_CONFIG register is set.

30.3.7 SNOR_DMA_CFG_0

SNOR DMA Configuration Register

Offset: 0x20 | Read/Write: R/W | Reset: 0x04000000 (0b000001000xxxxxxx00000000000000xx)

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

Bit	Reset	Description
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 0 = DISABLE 1 = ENABLE
15:2	0x0	WORD_COUNT: Specifies the number of words that need to be transferred.

30.3.8 SNOR_TIMING2_0

Tap Delay programming for the Trimmer on the Input path.

Offset: 0x28 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx000100)

Bit	Reset	Description
5:0	0x4	SNOR_IN_TAP_DELAY: Tap delay for the trimmer on the input path.

31.0 SATA CONTROLLER

31.1 Overview

The Serial Advance Technology Attachment (SATA) controller enables a control path from a Tegra® K1 device to an external SATA device. A SSD/HDD/ODD/e-SATA drive can be connected. This is a one port configuration.

The SATA controller is compliant with SATA specification rev. 3.1 and AHCI specification rev. 1.3.1.

31.2 Device ID

The device ID for the SATA controller in Tegra K1 is 0x0E1E. The AHCI driver needs to make sure the class code is set to 0x0106 and the program interface code is set to 0x01 before it starts operating the controller in AHCI mode.

31.3 Device Sleep

The SATA device sleep feature allows the host to put the SATA device into the low-power state by asserting the DEVSLP sideband signal in the following conditions:

- Software initiating DEVSLP only when the link is in slumber state
- Software initiating DEVSLP from any link state when the link is idle
- Hardware initiating DEVSLP only when the link is in slumber state and HW's idle timer times out
- Hardware initiating DEVSLP from any link state when the link is idle and HW's idle timer times out

The SATA design implements the control/status registers, timers, and revised port state machine with the additional PM.DevSleep state to support all four DEVSLP conditions as described in the TP001 to AHCI rev. 1.3.1 specification. The SATA controller implements the override bits in the Miscellaneous Register to allow software programming of the capability declarations.

31.3.1.1 Device Sleep and Unit Low Power State

After the link enters the DEVSLP state, the PAD should be put to IDDQ state, the PLL should be powered down, and the SATA controller itself can be put to either controller clock gating or ELPG.

Exiting from DEVSLP state can only be initialized by software; thus software should ensure the PAD, PLL, and SATA logic are restored to the operational state before triggering DEVSLP exit.

Since SATA can power gated when the link is in DEVSLP state, an override should be added to the Misc Register unit to override the DEVSLP sideband signal. The override should be set to keep the DEVSLP sideband asserted when SATA is power gated and the link is in DEVSLP state. The override should be cleared after SATA exited power gated.

The following figure illustrates the programming sequence of the DEVSLP with SATA entering and exiting power gating.

T1: SATA asserts DEVSLP; SATA_AUX asserts interrupt as a result of SATA asserting DEVSLP.

T2: SW clears SATA_AUX interrupt and starts programming sequence to put SATA to power gating, including save the context of SATA registers.

T3: SW sets SATA_AUX DEVSLP override.

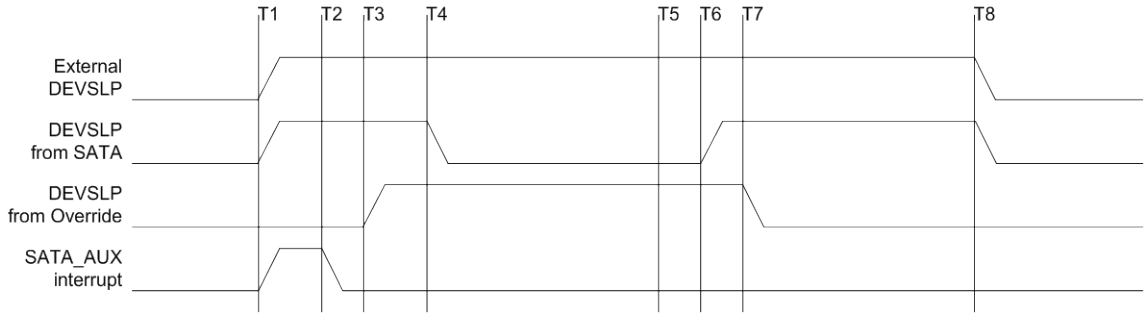
T4: SW removes power from SATA's power rail. Consequently, DEVSLP from SATA deasserts.

T5: SW restores power to SATA's power rail and starts programming sequence to restore context to SATA registers.

T6: SATA asserts its DEVSLP as a result of context restore.

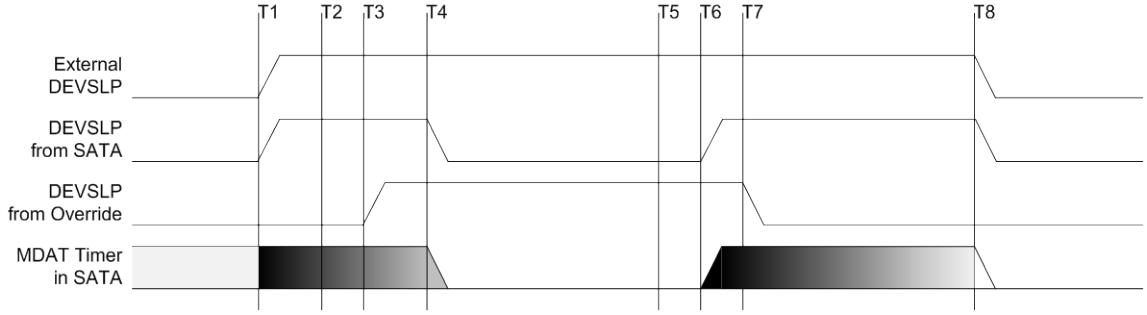
T7: SW clears SATA_AUX DEVSLP override.

T8: SATA deasserts DEVSLP.

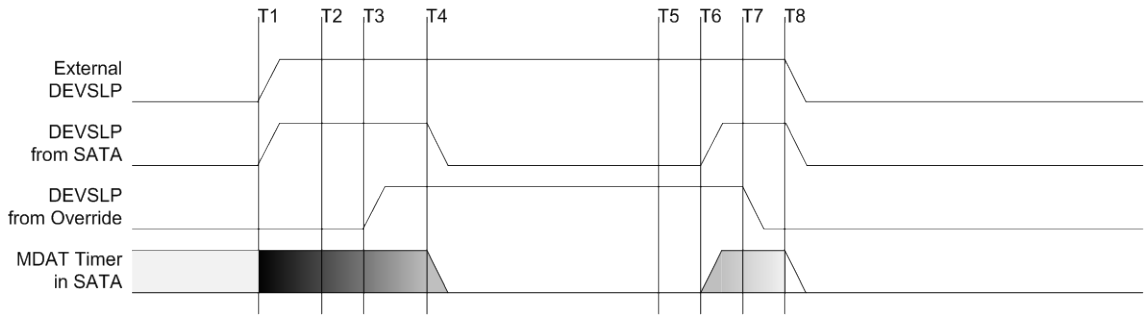


Update/Restore MDAT during ELPG

The MDAT timer in SATA is used to ensure DEVSLP is asserted for MDAT before it can be deasserted. As the following figure illustrates, after context is restored, the MDAT timer would be reloaded with the value programmed in PxDEVSLP.MDAT and counting down from there. This means DEVSLP would not be deasserted until MDAT time after context restore.



But to the SATA device, DEVSLP is kept asserted due to the override in SATA_AUX. Thus, it is possible to reduce the time SATA asserts DEVSLP after exited power gating by modify the value restored to MDAT timer, as the following figure illustrates.



The MDAT timer value should be updated/restored through MDAT_TIMER_AFTER_PG_VALID and MDAT_TIMER_AFTER_PG. It is the responsibility of software to track the time SATA is power gated and subtract it from the saved MDAT timer value before updating the value via the override.

31.3.1.2 Device Sleep with Port Multiplier

According to SATA I/O, device sleep feature is not used when the host controller is connecting to port multipliers:

31.3.2 AUX Version of PAD Idle Detection

In Tegra K1, the PAD supports VAUX bus idle detection. The Miscellaneous Register unit has been modified to support both the VCORE and VAUX version of bus idle detection for detecting the OOB signaling, while adding the control to put either or both the VCORE portion or the VAUX portion of the PAD in IDDQ mode when the SATA controller is power gated under the following conditions:

Link State	VCORE	VAUX	Note
Partial	ON	ON	VCORE portion is powered to maintain the common mode voltage. OOB detection using VCORE version with VAUX OOB detection disabled.
Slumber	OFF	ON	VAUX portion is used for OOB detection.
DEVSLP	OFF	OFF	OOB detection is not required during DEVSLP.

During LP0, the MUX between XUSB and SATA would keep the VAUX portion of the PAD in IDDQ, while the VCORE portion enters IDDQ when the system VCORE power rail is removed.

31.4 Power Gating Sequence

This section describes all the steps needed to achieve power gating.

- 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

31.4.1 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 software 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 hardware wake up is to be supported (for example, hot plug detection), the pads should be left in slumber. If hardware 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.

31.4.2 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 software needs to read for the interrupt.

1. To log the status and send an interrupt to software (software controlled PG) or signal to the PMC (hardware 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 software.

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 software clears the interrupt and the mask during normal operation.

31.4.3 Bits Required in Power Gating Sequences

The following bits are required in power gating sequences. Some of them are needed in both hardware and software 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 software 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.

31.4.4 Software Initiated Power Gating / Ungating 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 hardware 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 software to read a register. Ideally, this should be done only once.
4.	Driver needs to read and set the rx_idle_t registers in the APBMISC block 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 software 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.	The 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:* The driver shall write a bit in the Interrupt Controller 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 + interrupt controller) takes over interrupt generation.
8.	*If HW wake up (example hot-plug or async notification) is needed:* The driver shall place the SATA PADPLL in reset. *If Hw wake up is not needed:* Driver 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	The driver controls SATA PLL at run time (no HW low power sequencing)..
9.	The driver reports that SATA is being powered down. If PCIE is not being used, the driver 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	Wait Time= 1 µs
10.	The driver 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 hardware is such that it shall place the SATA PAD PLL in IDDQ mode but it is overridden by software value as swctl on iddq/reset is enabled above.
11.	The driver indicates to the AHCI software, that power gating is complete.		This is more of an implicit step; returning of the PG API function indicates that power-gating is done.

Steps involved in Power Ungating

#	Description	Register Programming	Comments
1	If hardware wake up is allowed and an OOB sequence is detected, an rx_stat interrupt will be sent by the interrupt controller 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 are in IDDQ mode, driver brings them 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	The driver checks if PLLE is already turned on or not. Part (a) : If PLLE is already turned on, it waits for 1 ms 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 bits. Part (a): If both bits are set, then the PLLE is already turned on. Wait for 1 ms . Part (b) If neither bit is 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 Reset values) – set the following values: CLK_RST_CONTROLLER_PLLE_BASE register: PLLE_PLDIV_CML = 0xd PLLE_PLDIV = 0x18 PLLE_MDIV = 0x1 PLLE_NDIV = 0xc8 CLK_RST_CONTROLLER_PLLE_MISC register PLLE_SETUP = 0x7 (This is not the default) PLLE_LOCK_ENABLE = ENABLE (1) (This isn’t default) 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 Enable the PLLE now by setting PLLE_ENABLE and PLLE_ENABLE_CML to ENABLE (1) in CLK_RST_CONTROLLER_PLLE_BASE register.	

#	Description	Register Programming	Comments
		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 = 0x25, CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_SSCINCINTRV = 0x20, 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	The driver 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	The driver toggles the power-gate SAX bit in the PMC to power-ungate SAX partition.	Toggle the SAX partition register bits in the PMC APBDEV_PMC_PWRGATE_TOGGLE_0.SAX=0 APBDEV_PMC_REMOVE_CLAMPING_CMD_0.SAX=1	
6	The power ungating is complete.		This is an implicit step. No action here.
7	Driver turns on the clocks to SATA and deasserts resets.	Software to ungate txclk and all SATA clocks, reset CLK_RST_CONTROLLER_CLK_OUT_ENB_V_0.SATA_CLKEN= ‘1’ , 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	Expected to take ~10 µs.

#	Description	Register Programming	Comments
		or not.	
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 hardware based low power sequencing.	SATAPAD_PLL_PLLE_IDDQ_RESET_SWCTL =0	
12	Driver indicates to the controller that the power ungating 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, the CPU goes to reset vector handler code in OS kernel patch which will restore the CPU state, reinitialize the CPU, un-powergate SATA, restore the SATA state, and give control to AHCI software.

If the CPU is powered up, the rx_stat interrupt signal generates a CPU interrupt, interrupt handler in driver shall un-powergate SATA and restore SATA state.

31.4.5 Software Changes

In software initiated power gating and ungating, software controls the entry and exit of the power gating controller. In the hardware initiated wake up mode, the software needs to read the bit oob_wake_detected to determine if the controller needs to be power ungated. The software needs to keep a backup of all the registers in system memory and needs to reprogram them once in the power ungating process. The software needs to program the registers in the correct sequence as mentioned above.

The PxCMD.CR bit 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 software initiating power-gating. To prevent accidental COMRESET from being sent on the SATA link, software needs to set an override for hardware not to look at SATA PLL lockdet when shutting down the SATA PLL. This can be done by writing SATA ext. config space register T_SATA<0>_CFG_MISC (0x550): write bit [10] to '1' and bit [8] to '0' (field T_SATA<0>_CFG__PHY_RESET_USAGE_MODE).

31.5 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 K1 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 “Address Translation Table” subsection. Even though, BAR 0 to Bar 4 are not applicable to Tegra K1 devices, they too have been allocated address ranges.

31.5.1 Address Translation Table

The next table shows how all of the SATA register spaces and SATA-IPFS registers are mapped to the Tegra K1 memory map. In Tegra K1 devices, 64KB APB space is allocated for all these registers and 512B APB space is 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

Register Section		DFPCI address [39:0]	APB adr[31:0] for DFPCI mapping	PRI address [31:0]	APB adr[31:0] for PRI mapping
	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
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 K1 devices, there is a standalone APB client created for SATA IOBIST register access, the PRI interface mapping from the SATA APB client is no longer valid					
The SATA IOBIST registers are mapped to 0x7000_6C00 - 0x7000_6DFF					

31.6 Programming Guidelines

31.6.1 Cold Boot

The IOPHY of SATA is shared with XUSB. Refer to the USB Complex section in this TRM for the IOPHY PLL initialization sequence.

The IOPHY lane ownership assignment is located in XUSB PADCTL, where the PAD parameter programming registers have also been moved to XUSB PADCTL. Refer to the USB Complex section in this TRM for the IOPHY PAD initialization sequence.

The PAD driver assigns the GPIO pins to the controllers:

- Set the following PINMUX register bits to initialize the GPIO pins for DEVSLP:
  - PINMUX_AUX_DAP_MCLK1_REQ_0[PM] to ‘SATA’
  - PINMUX_AUX_DAP_MCLK1_REQ_0[PUPD] to ‘NORMAL’
- Set the following PINMUX register bits to initialize the GPIO pins for DA:
  - PINMUX_AUX_GPIO_PFF2_0[PM] to ‘SATA’
  - PINMUX_AUX_GPIO_PFF2_0[PUPD] to ‘NORMAL’

The PAD driver programs the clocks and deasserts the resets to the controllers:

- Set the following CAR register bits to ‘1’ to enable the clocks to SATA:
  - CLK_RST_CONTROLLER_CLK_ENB_V_SET_0[SET_CLK_ENB_SATA]
  - CLK_RST_CONTROLLER_CLK_ENB_V_SET_0[SET_CLK_ENB_SATA_OOB]
- Program the following CAR register bits to set the source of the SATA clocks, where PLLP_OUT0 runs at 408 MHz:
  - CLK_RST_CONTROLLER_CLK_SOURCE_SATA_0[SATA_CLK_SRC] to ‘PLLP_OUT0’
  - CLK_RST_CONTROLLER_CLK_SOURCE_SATA_0[SATA_CLK_DIVISOR] to ‘0x6’
  - CLK_RST_CONTROLLER_CLK_SOURCE_SATA_OOB_0[SATA_OOB_CLK_SRC] to ‘PLLP_OUT0’
  - CLK_RST_CONTROLLER_CLK_SOURCE_SATA_OOB_0[SATA_OOB_CLK_DIVISOR] to ‘0x2’
- Set the following CAR register bits to ‘0’ to deassert reset to SATA:
  - CLK_RST_CONTROLLER_RST_DEVICES_V_0[SWR_SATA_RST]
  - CLK_RST_CONTROLLER_RST_DEVICES_V_0[SWR_SATA_OOB_RST]

The PAD driver brings the IOPHY out of IDDQ:

- Program the following XUSB PADCTL registers to ‘1’ to bring SATA IOPHY out of IDDQ.
  - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_SATA_PAD_IDDQ_DISABLE_MASK0]

The SATA PEP driver initializes the IPFS registers:

- Program the following SATA IPFS registers to allow software accesses to the SATA’s MMIO registers:
  - SATA_AXI_BAR5_START_0[AXI_BAR0_START] to ‘0x70020’
  - SATA_AXI_BAR5_SZ_0[AXI_BAR0_SIZE] to ‘0x00008’
  - SATA_FPCI_BAR5_0[FPCI_BAR0_START] to ‘0x0010000’
  - SATA_FPCI_BAR5_0[FPCI_BAR0_ACCESS_TYPE] to ‘0’
- Program the following SATA IPFS register to enable the SATA:
  - SATA_CONFIGURATION_0[EN_FPCI] to ‘1’

The SATA PEP driver initializes SATA’s configuration registers.

- Program the following SATA configuration registers to initialize SATA:
  - T_SATA<0>_CFG_1_BUS_MASTER to '1'
  - T_SATA<0>_CFG_1_MEMORY_SPACE to '1'
  - T_SATA<0>_CFG_9_BASE_ADDRESS to '0x08000'

The SATA PEP driver initializes SATA's AUX registers.

- Program the following SATA AUX registers to set the RX idle detection source and disable RX idle detection interrupt:
  - SATA_AUX_MISC_CTRL_1_0[AUX_OR_CORE_IDLE_STATUS_SEL] to 'CORE'
  - SATA_AUX_RX_STAT_INT_0[SATA_RX_STAT_INT_DISABLE] to '1'

31.6.2 ELPG Entry

The SATA PEP driver performs context saves for SATA registers specified in Section 31.7.1 .

The SATA PEP driver switches PAD control to SATA's AUX registers:

- Program the following SATA AUX registers to set the RX idle detection source and enable RX idle detection interrupt:
  - SATA_AUX_MISC_CTRL_1_0[AUX_OR_CORE_IDLE_STATUS_SEL] to 'AUX'
  - SATA_AUX_RX_STAT_INT_0[SATA_RX_STAT_INT_DISABLE] to '1'
- Program the following XUSB PADCTL registers to put the Vcore side of the IOPHY to IDDQ:
  - XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_1_0[IDDQ_OVRD] to '1'
  - XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_1_0[IDDQ] to '1'
- Check the assertion status of DEVSLP and set the DEVSLP override with the following SATA_AUX registers accordingly.
  - Read SATA_AUX_RX_STAT_INT_0[SATA_DEVLSP] as '$DEVSLP'
  - Set SATA_AUX_MISC_CTRL_1_0[DEVSLP_OVERRIDE] to '$DEVSLP'
- If DEVSLP# is asserted, read and store the MDAT timer value via the following *SATA* AUX register:
  - Read SATA_AUX_SPARE_CFG0_0[MDAT_TIMER_BEFORE_PG] as '$MDAT'

The System Power Management driver asserts reset to SATA then disables its clocks:

- Set the following CAR register bits to '1' to assert reset to SATA:
  - CLK_RST_CONTROLLER_RST_DEVICES_V_0[SWR_SATA_RST]
  - CLK_RST_CONTROLLER_RST_DEVICES_V_0[SWR_SATA_OOB_RST]
- Set the following CAR register bits to '1' to disable the clocks to individual SATA partitions:
  - CLK_RST_CONTROLLER_CLK_ENB_V_CLR_0[CLR_CLK_ENB_SATA]
  - CLK_RST_CONTROLLER_CLK_ENB_V_CLR_0[CLR_CLK_ENB_SATA_OOB]

The System Power Management driver uses software overrides to power down the IOPHY/UPHY PLL.

- Set the following CAR register bits to '1' to power down the PLL:
  - CLK_RST_CONTROLLER_SATA_PLL_CFG0_0[SATA_SEQ_PADPLL_PD_INPUT_VALUE]
  - CLK_RST_CONTROLLER_SATA_PLL_CFG0_0[SATA_SEQ_LANE_PD_INPUT_VALUE]
  - CLK_RST_CONTROLLER_SATA_PLL_CFG0_0[SATA_SEQ_RESET_PD_INPUT_VALUE]

The System Power Management driver disables the SATA power rails:

- Program the following PMC register bits in a single write to disable the power rail to SATA:

- APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
- APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘SAX’

- Read the following *PMC* register bits to confirm the power gating status of *SATA*:
  - APBDEV_PMC_PWRGATE_STATUS_0[SAX] equals ‘OFF’

31.6.3 ELPG Exit

The System Power Management driver uses software overrides to enable the IOPHY/UPHY PLL.

- Set the following *CAR* register bits to ‘0’ to enable the PLL
  - CLK_RST_CONTROLLER_SATA_PLL_CFG0_0[SATA_SEQ_PADPLL_PD_INPUT_VALUE]
  - CLK_RST_CONTROLLER_SATA_PLL_CFG0_0[SATA_SEQ_LANE_PD_INPUT_VALUE]
  - CLK_RST_CONTROLLER_SATA_PLL_CFG0_0[SATA_SEQ_RESET_PD_INPUT_VALUE]

The System Power Management driver enables the SATA power rails:

- Program the following PMC register bits in a single write to disable the power rail to SATA:
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘SAX’
- Read the following PMC register bits to confirm the power gating status of SATA:
  - APBDEV_PMC_PWRGATE_STATUS_0[SAX] equals ‘ON’

The System Power Management driver enables the SATA clocks:

- Set the following CAR register bits to ‘1’ to enable the clocks to the SATA partition:
  - CLK_RST_CONTROLLER_CLK_ENB_V_SET_0[SET_CLK_ENB_SATA]
  - CLK_RST_CONTROLLER_CLK_ENB_V_SET_0[SET_CLK_ENB_SATA_OOB]

The System Power Management driver removes power clamps to XUSB partitions:

- Set the following PMC register bits to ‘1’ to remove the power clamps to the SATA partitions:
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [SAX]
- Read the following PMC register bits to confirm the power clamps to the SATA partition are removed:
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [SAX] equals ‘0’

The System Power Management driver deasserts reset to SATA.

- Set the following CAR register bits to ‘0’ to assert reset to SATA:
  - CLK_RST_CONTROLLER_RST_DEVICES_V_0[SWR_SATA_RST]
  - CLK_RST_CONTROLLER_RST_DEVICES_V_0[SWR_SATA_OOB_RST]

The SATA PEP driver performs context restores for SATA registers specified in Section 31.7.1 ; starting from the IPFS registers followed by the SATA configuration registers and finally the SATA MMIO registers.

The SATA PEP driver switches PAD control to SATA:

- If DEVSLP is asserted, restore the MDAT timer value then deassert DEVSLP via the following *SATA* AUX registers:
  - Set SATA_AUX_SPARE_CFG0_0[MDAT_TIMER_AFTER_PG] to ‘$MDAT’
  - Set SATA_AUX_SPARE_CFG0_0[MDAT_TIMER_AFTER_PG_VALID] to ‘1’
  - Set SATA_AUX_MISC_CTRL_1_0[DEVSLP_OVERRIDE] to ‘0’
- Program the following XUSB_PADCTL registers to bring the Vcore side of the IOPHY out of IDDQ:

- XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_1_0[IDDQ_OVRD] to ‘0’
- XUSB_PADCTL_IOPHY_MISC_PAD_S0_CTL_1_0[IDDQ] to ‘0’

- Program the following SATA_AUX registers to set the RX idle detection source, disable RX idle detection interrupt, and clear RX idle interrupt:
  - SATA_AUX_MISC_CTRL_1_0[AUX_OR_CORE_IDLE_STATUS_SEL] to ‘CORE’
  - SATA_AUX_RX_STAT_INT_0[SATA_RX_STAT_INT_DISABLE] to ‘0’
  - SATA_AUX_RX_STAT_CLR_0[SATA_RX_STAT_INT_CLR] to ‘1’

31.6.4 LP0

SATA AUX power is also removed during LP0, thus the assertion status of DEVSLP# through LP0 should be maintained by using the pull-down resistor of the GPIO pad via the PINMUX_AUX_DAP_MCLK1_REQ_0[PUPD] register:

If DEVSLP is asserted, the PAD driver should enable the pull-up of the GPIO pin by programming the following register before entering LP0.

- PINMUX_AUX_DAP_MCLK1_REQ_0[PUPD] to ‘PU’

If DEVSLP is asserted, the PAD driver should disable the pull-up of the GPIO pin by programming the following register after exiting LP0 and the DEVSLP override in SATA AUX register has been set:

- PINMUX_AUX_DAP_MCLK1_REQ_0[PUPD] to ‘NORMAL’

As part of the LP0 entry sequence, after setting the SATA controller to ELPG, the PAD driver should put the IOPHY to IDDQ:

- Program the following XUSB PADCTL registers to ‘0’ to bring SATA IOPHY to IDDQ.
  - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_SATA_PAD_IDDQ_DISABLE_MASK0]

As part of the LP0 exit sequence, after restoring power to SATA, the PAD driver brings the IOPHY out of IDDQ:

- Program the following XUSB PADCTL registers to ‘1’ to bring SATA IOPHY out of IDDQ:
  - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_SATA_PAD_IDDQ_DISABLE_MASK0]

31.7 Registers

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

31.7.1 Registers for Save and Restore

The registers needed to be saved and restored are:

IPFS registers

Save and restore	Offset (hex)	Save and restore	Offset (hex)
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

Save and restore	Offset (hex)	Save and restore	Offset (hex)
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

Note: Registers present in both the SATA and SATA0 address spaces are listed as SATA<0>.

Save and restore	Offset (hex)	Save and restore	Offset (hex)
T_SATA<0>_CFG_1	4	T_SATA<0>_CFG_LINK_0	174
T_SATA<0>_CFG_3	C	T_SATA<0>_CFG_LINK_1	178
T_SATA<0>_CFG_9	24	T_SATA<0>_CFG_LINK_2	17c
T_SATA<0>_CFG_10	28	MCP_SATA<0>_CFG_TRANS_0	1D0
T_SATA<0>_CFG_12	30	T_SATA0_ALPM_CTRL	238
T_SATA<0>_CFG_13	34	T_SATA0_AHCI_HBA_CTL_0	30C
T_SATA<0>_CFG_14	38	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL	318
T_SATA<0>_CFG_15	3C	T_SATA0_AHCI_HBA_SPARE_1	320
T_SATA<0>_CFG_16	40	T_SATA0_AHCI_HBA_SPARE_2	324
T_SATA<0>_CFG_17	44	T_SATA0_AHCI_HBA_DYN_CLK_CLAMP	328
T_SATA<0>_CFG_18	48	T_SATA0_AHCI_CFG_ERR_CTRL	32C
T_SATA<0>_MSI_CTRL	B0	T_SATA0_AHCI_HBA_CTL_1	338
T_SATA<0>_MSI_ADDR1	B4	T_SATA0_AHCI_HBA_PRE_STAGING_CONTROL	340
T_SATA<0>_MSI_ADDR2	B8	T_SATA0_CFG_FPCI_0	430
T_SATA<0>_MSI_DATA	BC	T_SATA0_CFG_ESATA_CTRL	494
T_SATA<0>_MSI_QUEUE	C0	T_SATA<0>_CTL1	4A0
T_SATA<0>_MSI_MAP	EC	T_SATA<0>_CFG_CTL_GLUE	4B0
T_SATA0_CFG_PHY_POWER	124	T_SATA<0>_PHY_CTRL	534
T_SATA0_CFG_PHY_POWER_1	128	T_SATA<0>_CTRL	540
T_SATA<0>_CFG_PHY_1	12C	T_SATA<0>_LOW_POWER_COUNT	554

Save and Restore Following Protocol

(Program T_SATA<0>_INDEX to select port 0)

Save and restore	Offset (hex)	Save and restore	Offset (hex)
T_SATA<0>_CHXCFG1	530	T_SATA<0>_CHX_PHY_CTRL_3	6B0
T_SATA<0>_CHX_MISC	684	T_SATA<0>_CHX_PHY_CTRL_4	6B4
T_SATA<0>_CHXCFG3	700	T_SATA<0>_CHX_PHY_CTRL_5	6B8
T_SATA<0>_CHXCFG4_CHX	704	T_SATA<0>_CHX_PHY_CTRL_6	6BC
T_SATA<0>_CHX_PHY_CTRL1_GEN1	690	T_SATA<0>_PRBS_CHX – OP	714
T_SATA<0>_CHX_PHY_CTRL1_GEN2	694	T_SATA<0>_CHX_LINK0	750
T_SATA<0>_CHX_PHY_CTRL1_GEN3	698	T_SATA<0>_CHX_GLUE	7F0

Save and restore	Offset (hex)	Save and restore	Offset (hex)
T_SATA<0>_CHX_PHY_CTRL_2	69C		

BAR 5 Space Registers

Save and restore	Offset (hex)	Save and restore	Offset (hex)
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

Power-Gating Registers

These registers require save and restore across the power-gating sequence. Apart from this, the software is expected to save and restore the configuration space registers.

Save and restore	Offset (hex)	Save and restore	Offset (hex)
T_AHCI_HBA_SPARE_0	a4	T_AHCI_HBA_CCC_PORTS	18
T_AHCI_HBA_GHC	4	T_AHCI_HBA_CCC_CTL - OP (optional)	14
T_AHCI_HBA_EM_LOC	1c	T_AHCI_HBA_EM_CTL - OP	20
T_AHCI_PORT_PXCLB	100	T_AHCI_PORT_PXCLBU	104
T_AHCI_PORT_PXFB	108	T_AHCI_PORT_PXFBU	10c
T_AHCI_PORT_PXIE	114	T_AHCI_PORT_PXCMD	118
T_AHCI_HBA_SHUTDOWN_TIMER	ac	T_AHCI_HBA_PLL_CTRL	a8
T_AHCI_PORT_PXSCTL	12c	T_AHCI_PORT_PXFBS	140
T_AHCI_PORT_MP	17c	T_AHCI_PORT_PXDEVSLP	144

Save and Restore via Backdoor Writes

Save and restore	Offset (hex)
T_AHCI_HBA_CAP_0 (T_SATA0_AHCI_HBA_CAP_BKDR)	0 (300*)
T_AHCI_HBA_PI (T_SATA0_AHCI_HBA_PI_BKDR)	0C (33C*)
T_AHCI_HBA_CAP2 (T_SATA0_AHCI_HBA_CAP2_BKDR)	24 (330*)
T_AHCI_PORT_PXTFD (T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR)	120 (790*)
T_AHCI_PORT_PXSIG (T_SATA<0>_CHX_AHCI_PORT_PXSIG_BKDR)	124 (794*)
T_AHCI_PORT_PXSSTS (T_SATA<0>_CHX_AHCI_PORT_PXSSTS_BKDR)	128 (798*)

Note: * Backdoor addresses 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

31.7.2 Registers Used in Power Gating/Ungating Sequence

Refer to the PMC and Clock and Reset Controller sections for the details of these registers:

- APBDEV_PMC_SATA_PWRGT_0

- CLK_RST_CONTROLLER_SATA_PLL_CFG0_0
- CLK_RST_CONTROLLER_PLLE_AUX_0

31.7.3 SATA Register Descriptions

31.7.3.1 SATA_AXI_BARi_SZ_0 Registers

There are eight SATA_AXI_BARi_SZ_0 registers (I = 0 through 7). The size of the address range associated with BARi is in 4K increments. A value of 0 signifies BARi is not used.

SATA_AXI_BAR0_START_0 through SATA_AXI_BAR4_START_0

- For BAR0 through BAR4 (i = 0 through 4):

Offset: 0x0 + (i * 0x4) | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000001)

Bit	Reset	Description
19:0	0x1	AXI_BARi_SIZE

SATA_AXI_BAR5_START_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000002 (0b00000000000000000010)

Bit	Reset	Description
19:0	0x2	AXI_BAR5_SIZE

SATA_AXI_BAR6_START_0 through SATA_AXI_BAR7_START_0

- For BAR6 through BAR7 (i = 6 through 7)

Offset: 0x18 + (i * 0x4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000)

Bit	Reset	Description
19:0	0x0	AXI_BARi_SIZE

31.7.3.2 SATA_AXI_BARi_START_0 Registers

There are eight SATA_AXI_BARi_START registers. The AXI target address is compared to the start/size for each BAR to determine if the access is to that BAR.

SATA_AXI_BAR0_START_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x70022000 (0b01110000000000100010)

Bit	Reset	Description
31:12	0x70022	AXI_BAR0_START:The start of AXI address space for BAR0.

SATA_AXI_BAR1_START_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x70023000 (0b01110000000000100011)

Bit	Reset	Description
31:12	0x70023	AXI_BAR1_START:The start of AXI address space for BAR1.

SATA_AXI_BAR2_START_0

Offset: 0x48 | Read/Write: R/W | Reset: 0x70024000 (0b01110000000000100100)

Bit	Reset	Description
31:12	0x70024	AXI_BAR2_START:The start of AXI address space for BAR2.

SATA_AXI_BAR3_START_0

Offset: 0x4c | Read/Write: R/W | Reset: 0x70025000 (0b01110000000000100101)

Bit	Reset	Description
31:12	0x70025	AXI_BAR3_START:The start of AXI address space for BAR3.

SATA_AXI_BAR4_START_0

Offset: 0x50 | Read/Write: R/W | Reset: 0x70026000 (0b01110000000000100110)

Bit	Reset	Description
31:12	0x70026	AXI_BAR4_START: The start of AXI address space for BAR4.

SATA_AXI_BAR5_START_0

Offset: 0x54 | Read/Write: R/W | Reset: 0x70027000 (0b01110000000000100111)

Bit	Reset	Description
31:12	0x70027	AXI_BAR5_START: The start of AXI address space for BAR5.

SATA_AXI_BAR6_START_0

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000)

Bit	Reset	Description
31:12	0x0	AXI_BAR6_START: The start of AXI address space for BAR6.

SATA_AXI_BAR7_START_0

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000)

Bit	Reset	Description
31:12	0x0	AXI_BAR7_START: The start of AXI address space for BAR7.

31.7.3.3 SATA_FPCI_BARi_0 Registers

There are eight SATA_FPCI_BARi_0 registers. All registers have these two fields:

- Bits [31:4]: FPCI_BARi_START: This field contains the start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a shift left 12 of the value of this field.
- Bit 0: FPCI_BARi_ACCESS_TYPE: This field indicates if the address region is memory mapped versus configuration or I/ O space.
  0 = Memory-mapped access (PW only)
  1 = I/O / config access (NPW only)

SATA_FPCI_BAR0_0 through SATA_FPCI_BAR5_0

For BAR0 through BAR5 (i = 0 through 5)

Offset: 0x80 + (i * 0x4) | Read/Write: R/W | Reset: 0xfdfc0001 (0b11111101111111100000000000000xxx1)

Bit	Reset	Description
31:4	0xfdfc000	FPCI_BARi_START
0	0x1	FPCI_BARi_ACCESS_TYPE

SATA_FPCI_BAR6_0 through SATA_FPCI_BAR7_0

Offset: 0x98 + (i* 0x4) | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000000000000000xxx1)

Bit	Reset	Description
31:4	0x0	FPCI_BARi_START
0	0x1	FPCI_BARi_ACCESS_TYPE

31.7.3.4 SATA_MSI_BAR_SZ_0

MSI BAR Size

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000)

Bit	Reset	Description
19:0	0x0	MSI_BAR_SIZE: The size of the address range associated with MSI BAR is in 4K increments. A value of 0 signifies MSI BAR is not used.

31.7.3.5 SATA_MSI_AXI_BAR_ST_0

MSI AXI BAR Start

Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000)

Bit	Reset	Description
31:12	0x0	MSI_AXI_BAR_START: The start of upstream AXI address space for MSI BAR. The upstream FPCI address is compared to start/1KB range for MSI BAR to determine if the access is MSI. Bits 31:12 of MSI BAR start correspond to AXI address bits 31:12.

31.7.3.6 SATA_MSI_FPCI_BAR_ST_0

MSI FPCI BAR Start

Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:4	0x0	MSI_FPCI_BAR_START: The start of upstream FPCI address space for MSI BAR. The upstream FPCI address is compared to start/1KB range for MSI BAR to determine if the access is MSI. Bits 31:4 of MSI BAR start correspond to UFPCI address bits 39:12.

31.7.3.7 SATA_MSI_VECi_0 Registers

There are eight SATA_MSI_VECi_0 vector registers (i = 0 through 7). 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.

Offset: 0x100 + (i * 0x4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTORi

31.7.3.8 SATA_MSI_EN_VEC0_0

There are eight SATA_MSI_EN_VECi_0 vector registers (i = 0 through 7). 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.

Offset: 0x140 + (i * 0x4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTORi

31.7.3.9 SATA_CONFIGURATION_0

Configuration

Offset: 0x180 | Read/Write: R/W | Reset: 0x800X8X40 (0b1xxxxxxxxxxx1xxx10xxxxxx01000000)

Bit	R/W	Reset	Description
31	RW	0x1	CLKEN_OVERRIDE: This can override the clock enable in case of malfunction.
19	RW	0x1	PW_NO_DEVSEL_ERR_CYA: Setting this bit will disable detection of DECERR due to no DEVSEL for DS PWs only.
18	RO	X	INITIATOR_READ_IDLE: This read-only bit provides status reads on the AFI upstream. A value of 1b indicates there are no outstanding reads to the initiator.
17	RO	X	INITIATOR_WRITE_IDLE: This read-only bit provides status writes on the AFI upstream. A value of 1b indicates there are no outstanding writes to the initiator.
15	RW	0x1	WDATA_LEAD_CYA: Used to enable/disable the handling of write data ahead of requests on the IPFS AXI target.
14	RW	0x0	WR_INTRLV_CYA: Used to enable/disable the handling of interleaved write requests on the IPFS AXI target
11	RO	X	TARGET_READ_IDLE: This read-only bit provides status reads to the IPFS target. A value of 1b indicates there are no outstanding reads to the downstream FPCI.
10	RO	X	TARGET_WRITE_IDLE: This read-only bit provides status writes to IPFS target. A value of 1b indicates there are no outstanding writes to the downstream FPCI.
9	RO	X	MSI_VEC_EMPTY: This read-only bit provides status on whether MSI Vector registers have any active bits valid or not.
7	RW	0x0	UFPCI_MSIAW: MSI After Write ordering rule. 1 = whenever MSI is ready assert the interrupt 0 = default behavior, apply MSIAW ordering rule
6	RW	0x1	UFPCI_PWPASSPW: Input to upstream FPCI 1 = whenever a write is ready, send it 0 = write goes only when outstanding PWs outside of new write's region are retired (default).
5	RW	0x0	UFPCI_PASSPW: Input to the upstream FPCI. Allow the upstream FPCI reads to pass writes.
4	RW	0x0	UFPCI_PWPASSNPW: Used for upstream FPCI. Allow upstream FPCI PWs to pass NPWs.

Bit	R/W	Reset	Description
3	RW	0x0	DFPCI_PWPASSNPW: Used for the downstream FPCI. Allow downstream FPCI PWs to pass NPWs.
2	RW	0x0	DFPCI_RSPPASSPW: Input to the downstream FPCI. Allow downstream FPCI responses to pass writes.
1	RW	0x0	DFPCI_PASSPW: Input to the downstream FPCI. Allow downstream FPCI reads to pass writes.
0	RW	0x0	EN_FPCI: When the IPFS device block is disabled, it is completely invisible on the IPFS bus; that is, it does not even process IPFS configuration accesses.

31.7.3.10 SATA_FPCI_ERROR_MASKS_0

FPCI Error Masks

Offset: 0x184 | Read/Write: R/W | Reset: 0x00000000 (0b000)

Bit	Reset	Description
2	0x0	MASK_FPCI_MASTER_ABORT: This bit allows an FPCI error to be forwarded to the AXI response when the FPCI error response indicates Master Abort. 1 = forward error 0 = return AXI OKAY response (2'b0)
1	0x0	MASK_FPCI_DATA_ERROR: This bit allows an FPCI error to be forwarded to the AXI response when the FPCI error response indicates Data Error. 1 = forward error 0 = return AXI OKAY response (2'b0)
0	0x0	MASK_FPCI_TARGET_ABORT: This bit allows an FPCI error to be forwarded to the AXI response when the FPCI error response indicates Target Abort. This bit also covers decode errors generated when there is no DEVSEL received. 1 = forward error 0 = return AXI OKAY response (2'b0)

31.7.3.11 SATA_INTR_MASK_0

Interrupt Masks

Offset: 0x188 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx0xxxxxxx0)

Bit	Reset	Description
16	0x0	IP_INT_MASK: IP (SATA/AZA) interrupt to the CPU gated by a mask.
8	0x0	MSI_MASK: MSI to the CPU gated by a mask.
0	0x0	INT_MASK: Interrupt to the CPU gated by a mask.

31.7.3.12 SATA_INTR_CODE_0

Interrupt Control

Offset: 0x18c | Read/Write: R/W | Reset: 0x00000000 (0b00000)

Bit	Reset	Description
4:0	0x0	INT_CODE: Interrupt codes. If the code is 0, logging of the next interrupt is enabled 0 = INT_CODE_CLEAR : Clear interrupt code 1 = INT_CODE_INI_SLVERR : Interrupt code for CPU AXI SLVERR response to IPFS 2 = INT_CODE_INI_DECERR : Interrupt code for CPU AXI DECERR response to IPFS 3 = INT_CODE_TGT_SLVERR : Interrupt code for PCIe endpoint FPCI target abort or data error response to IPFS 4 = INT_CODE_TGT_DECERR : Interrupt code for PCIe2 FPCI master abort response to IPFS 5 = INT_CODE_TGT_WRERR : Interrupt code for bufferable write to non-posted write address region

Bit	Reset	Description
		6 = RSVD1 : Reserved 7 = INT_CODE_DFPCI_DECERR : Interrupt code for PCIe2 response to downstream request when downstream FPCI address does not fall in a claimable downstream region 8 = INT_CODE_AXI_DECERR : Interrupt code for IPFS response to downstream request when AXI target address does not fall in any of IPFS downstream BARs 9 = INT_CODE_FPCI_TIMEOUT : Interrupt code for FPCI Timeout 10 = RSVD2 : Reserved for future expansion 11 = RSVD3 12 = RSVD4 13 = RSVD5 14 = RSVD6 15 = INT_CODE_SM_FATAL_ERROR : Interrupt code for SM fatal error 16 = INT_CODE_SM_NON_FATAL_ERROR : Interrupt code for SM non-fatal error

31.7.3.13 SATA_INTR_SIGNATURE_0

Interrupt Signature

Offset: 0x190 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000x0)

Bit	Reset	Description
31:2	0x0	INT_INFO: For interrupt codes 1-5/7-8, this field contains address bits [31:2], either in FPCI memory space or AXI space. For FPCI generated errors, this field contains the FPCI address. For AXI/IPFS generated errors, this field contains the AXI address.
0	0x0	DIR: Indicates the direction of the AXI/FPCI transaction. If the signature type is 6 (sideband message), this field is 1. 1=rd 0=wr 0 = WRITE : Interrupt due to a write transaction 1 = READ : Interrupt due to a read transaction

31.7.3.14 SATA_UPPER_FPCI_ADDR_0

Upper FPCI Address

Offset: 0x194 | Read/Write: R/W | Reset: 0x00000000 (0b00000000)

Bit	Reset	Description
7:0	0x0	INT_INFO_UPPER: These 8 bits are the upper byte of the captured FPCI address (bits [39:32])when the interrupt code is 3, 4, or 7.

31.7.3.15 SATA_IPFS_INTR_ENABLE_0

IPFS Interrupt Enable

Offset: 0x198 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxx00000000)

Bit	Reset	Description
13	0x0	EN_SM_NON_FATAL_ERROR: Enable bit for interrupt code 15
12	0x0	EN_SM_FATAL_ERROR: Enable bit for interrupt code 14
7	0x0	EN_FPCI_TIMEOUT: Enable bit for interrupt code 9
6	0x0	EN_AXI_DECERR: Enable bit for interrupt code 8
5	0x0	EN_DFPCI_DECERR: Enable bit for interrupt code 7
4	0x0	EN_TGT_WRERR: Enable bit for interrupt code 5

Bit	Reset	Description
3	0x0	EN_TGT_DECERR: Enable bit for interrupt code 4
2	0x0	EN_TGT_SLVERR: Enable bit for interrupt code 3
1	0x0	EN_INI_DECERR: Enable bit for interrupt code 2
0	0x0	EN_INI_SLVERR: Enable bit for interrupt code 1

31.7.3.16 SATA_UFPCI_CONFIG_0

Upstream FPCI Configuration

Offset: 0x19c | Read/Write: R/W | Reset: 0x00000002 (0b00010)

Bit	Reset	Description
4:0	0x2	UNITID_T0C0: Upstream FPCI Unit ID for controller 0. HyperTransport, upstream FPCI request

31.7.3.17 SATA_CFG_REVID_0

CFG_REVID register

Offset: 0x1a0 | Read/Write: R/W | Reset: 0x000X10XX (0bxxxxxx0100xxxxxxx1)

Bit	R/W	Reset	Description
19	RO	X	DEV2SM_NONISO_REQUEST_PEND: Indicates if a non ISO request is pending. 0 = NO 1 = YES
18	RO	X	DEV2SM_ISO_REQUEST_PEND: Indicates if an ISO request is pending. 0 = NO 1 = YES
13:12	RW	0x1	STRAP_CPU_MODE: MCP : Mode to send MSI. Can be programmable 0 = NB_INTEL 1 = NB_AMD 2 = AMD 3 = TMTA
11	RW	0x0	CFG_REVID_WRITE_ENABLE: MCP : Enable to override the rev ID. Can be programmable 0 = CLEAR 1 = SET
10	RW	0x0	CFG_REVID_OVERRIDE: MCP : Can override the current revision ID. Can be programmable 0 = DISABLE 1 = ENABLE
4	RO	X	DEV2LEG_NONCOH_REQUEST_PEND: MCP : Tells the leg block that a non-coherent request is pending. 0 = NO 1 = YES
3	RO	X	DEV2LEG_COH_REQUEST_PEND: MCP comment : Tells the leg block that a coherent request is pending 0 = NO 1 = YES
2	RW	0x1	SM2DEV_FPCI_TIMEOUT_EN: FPCI timeout enable bit for Controller 0 = DISABLE 1 = ENABLE

31.7.3.18 SATA_FPCI_TIMEOUT_0

FPCI_TIMEOUT register

Offset: 0x1a4 | Read/Write: R/W | Reset: 0x000f0000 (0b11110000000000000000)

Bit	Reset	Description
19:0	0xf0000	SM2ALL_FPCI_TIMEOUT_THRESH: This sets the timeout threshold value for the FPCI bus. The counter starts counting when each queue (iso/niso- rd/wr) has a pending request in the FPCI wrapper. The count resets when the requests are popped.

31.7.3.19 SATA_TOM_0

Top of Memory Limit

Offset: 0x1a8 | Read/Write: R/W | Reset: 0x3fff0fff (0b11111111111111xxxx111111111111)

Bit	Reset	Description
29:16	0x3fff	LEG2ALL_TOM2: Top of Memory Limit 2.
11:0	0xfff	LEG2ALL_TOM1: Top of Memory Limit 1.

31.7.3.20 SATA_INITIATOR_ISO_PW_RESP_PENDING_0

Initiator ISO PW Response Pending

Offset: 0x1ac | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxx)

Bit	Reset	Description
7:0	X	NUM_INITIATOR_ISO_PW_RESP_PEND: Number of pending initiator ISO PW responses

31.7.3.21 SATA_INITIATOR_NISO_PW_RESP_PENDING_0

Initiator Non-ISO PW Response Pending

Offset: 0x1b0 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxx)

Bit	Reset	Description
7:0	X	NUM_INITIATOR_NISO_PW_RESP_PEND: Number of pending initiator NISO PW responses

31.7.3.22 SATA_INTR_STATUS_0

IPFS Interrupt Status

Offset: 0x1b4 | Read/Write: RO | Reset: 0x0000000X (0bxxx)

Bit	Reset	Description
2	X	IP_INTR_STATUS: Status of IP (SATA/AZA) interrupt
1	X	MSI_INTR_STATUS: Status of MSI interrupt
0	X	IPFS_INTR_STATUS: Status of IPFS interrupt

31.7.3.23 SATA_DFPCI_BEN_0

Downstream FPCI Byte Enables

Offset: 0x1b8 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
31	0x0	EN_DFPCI_BEN: Enable bit for BEN; when set, programmed BE is sent on the DFPCI bus
3:0	0x0	DFPCI_BYTE_ENABLE_N: Active low byte enables

31.7.3.24 SATA_CLKGATE_HYSTERESIS_0

Offset: 0x1bc | Read/Write: R/W | Reset: 0x00000014 (0b00010100)

Bit	Reset	Description
7:0	0x14	CLK_DISABLE_CNT: Number of IPFS clock cycles to wait after clock gating criteria is met to disable IPFS/FPCI clocks

31.7.3.25 SATA_SATA_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control (where applicable) and Clock Gating Control Register.

Note: The FIFO timing aspects of this register are no longer supported, but retained for software compatibility

The clock override/ovr_mode fields of this register control the second-level clock gating for the client and the MC side of the MCCIF. All clock gating is enabled by default.

A '1' written to the rclk/wclk override field will result in one of the following:

- With wclk/rclk override mode = LEGACY, the clock reverts to legacy mode of operation where the clock is on whenever the client clock is enabled.
- With wclk/rclk override mode = ON, the clock is always on inside the MCCIF and PC. A '1' written to the cclk override field keeps the client's clock always on inside the MCCIF.

Offset: 0x1dc | Read/Write: R/W | Reset: 0x00000000 (0b00000xxxxxxxxxxxx0000)

Bit	Reset	Description
20	LEGACY	SATA_RCLK_OVR_MODE: 0 = LEGACY 1 = ON
19	LEGACY	SATA_WCLK_OVR_MODE: 0 = LEGACY 1 = ON
18	0x0	SATA_CCLK_OVERRIDE
17	0x0	SATA_RCLK_OVERRIDE
16	0x0	SATA_WCLK_OVERRIDE
3	DISABLE	SATA_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	SATA_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	SATA_MCCIF_RDMC_RDFAST: 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
0	DISABLE	SATA_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

31.7.3.26 SATA_ORDERING_RULES_0

Offset: 0x1e0 | Read/Write: R/W | Reset: 0x00000000 (0b000)

Bit	Reset	Description
3	0x0	UPSTREAM_MSIAW: Modified upstream MSIAW ordering. 0 = Tegra K1 MSIAW behavior 1 = Tegra 3 MSIAW behavior
2	0x0	UPSTREAM_RESPAW: Modified RespAW ordering. 0 = Tegra K1 RespAW behavior 1 = Tegra 3 RespAW behavior
1	0x0	UPSTREAM_RAW: Modified RAW ordering. 0 = Tegra K1 RAW behavior 1 = Tegra 3 RAW behavior

31.7.3.27 SATA_A2F_UFPCI_CFG0_0

Offset: 0x1e4 | Read/Write: R/W | Reset: 0x00000050 (0b00000000xxxxx0000000000001010000)

Bit	Reset	Description
31:24	0x0	STATIC_WAIT_IDLE_CNTR
18:16	0x0	STATIC_UFPCI_UFA_STARVE_CNTR_PRI1
15:12	0x0	STATIC_UFPCI_UFA_STARVE_CNTR_PRI0
11:10	0x0	STATIC_UFPCI_RR_BURST_SZ_PRI1
9:8	0x0	STATIC_UFPCI_RR_BURST_SZ_PRI0
7	0x0	STATIC_WAIT_CLAMP_EN
6	0x1	STATIC_UFPCI_UFA_DYN_BLOCK_EN
5	0x0	STATIC_UFPCI_UFA_BLK_COHERENT
4:2	0x4	STATIC_UFPCI_BLOCK_CMD_THRESHOLD
1	0x0	STATIC_CYA_UFA_ARB
0	0x0	STATIC_CYA_BACK2BACK_UPSTREAM_BLOCK

31.7.3.28 SATA_A2F_UFPCI_CFG1_0

Offset: 0x1e8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000)

Bit	Reset	Description
7:0	0x0	STATIC_WAIT_UNCLAMP_CNTR

31.7.4 SATA AHCI Register Descriptions

31.7.4.1 Generic Host Control Registers

T_AHCI_HBA_CAP_0

HBA_CAP (Host Bus Adaptor) Capabilities

This register indicates basic capabilities to software.

Offset: 0x00 | Read/Write: RO | Reset: 0xE620FF00

Bit	R/W	Reset	Description
31	R	1	T_AHCI_HBA_CAP_S64A: 1h: S64A_TRUE (default) 0h: S64A_FALSE
30	R	1	T_AHCI_HBA_CAP_SNCQ: 1h: SNCQ_TRUE (default) 0h: SNCQ_FALSE
29	R	1	T_AHCI_HBA_CAP_SSNTF: 1h: SSNTF_TRUE (default) 0h: SSNTF_FALSE
28	R	0	T_AHCI_HBA_CAP_SMPS: Supports Mechanical Presence Switch. 1h: SMPS_TRUE 0h: SMPS_FALSE (default)
27	R	0	T_AHCI_HBA_CAP_SSS: Supports Staggered Spin-up. When set to '1' the HBA supports staggered spin-up on its ports, for use in balancing power spikes. This needs to be set by BIOS. 1h: SSS_TRUE 0h: SSS_FALSE (default)
26	R	1	T_AHCI_HBA_CAP_SALP: Supports Aggressive Link Power Management. When set to '1' the HBA can support auto-generating link requests to the Partial or Slumber states when there are no commands to process. 1h: SALP_TRUE (default) 0h: SALP_FALSE
25	R	1	T_AHCI_HBA_CAP_SAL: Supports Activity LED. When set to '1' the HBA supports a single activity indication output pin. 1h: SAL_TRUE (default) 0h: SAL_FALSE
24	R	0	T_AHCI_HBA_CAP_SCLO: Supports Command List Override. When set to '1' HBA supports the T_AHCI_PORT_PXCMD_CLO nit and its associated functionality. When cleared to '0' the HBA is not capable of clearing the BSY and DRQ bits in the status register in order to issue a software reset if these bits are still set from a previous operation. 1h: SCLO_TRUE 0h: SCLO_FALSE (default)
23:20	R	2h	T_AHCI_HBA_CAP_ISS: Interface Speed Supported. Indicates the maximum interface speed supported by the ports. 0h: ISS_RSVD 1h: ISS_GEN1 2h: ISS_GEN1_2 (default)
19	R	0	T_AHCI_HBA_CAP_SNZO: Supports Non-Zero DMA Offsets. When set to '1' indicates that the HBA can support non-zero DMA offsets for DMA setup FISes. 1h: SNZO_TRUE 0h: SNZO_FALSE (default)
18	R	0	T_AHCI_HBA_CAP_SAM: Supports AHCI mode only. When set to '1' indicates that the SATA controller doesnot implement a legacy, task-file based register interface. When cleared indicates that controller supports legacy mechanism in addition to AHCI mode. 1h: SAM_TRUE 0h: SAM_FALSE (default)

Bit	R/W	Reset	Description
17	R	0	T_AHCI_HBA_CAP_SPM: Supports Port Multiplier. When set to '1' indicates that a port multiplier with command based switching is supported. 1h: SPM_TRUE 0h: SPM_FALSE (default)
16	R	0	T_AHCI_HBA_CAP_FBSS: FIS Based Switching Supported. When set to '1' indicates that the HBA supports Port Multiplier FIS-Based switching. 1h: FBSS_TRUE 0h: FBSS_FALSE (default)
15	R	1	T_AHCI_HBA_CAP_PMD: PIO Multiple DRQ Block. If set to '1' the HBA supports multiple DRQ block data transfers for the PIO command protocol. 1h: PMD_SUPPORTED (default) 0h: PMD_NOT_SUPPORTED
14	R	1	T_AHCI_HBA_CAP_SSC: Supports Slumber State. This field indicates HBA supports transitions to the slumber state to manage power. When set to 1 HBA and device initiated Slumber requests can be supported. 1h: SSC_TRUE (default) 0h: SSC_FALSE
13	R	1	T_AHCI_HBA_CAP_PSC: Supports Partial State. This field indicates HBA supports transitions to the partial state to manage power. When set to 1 HBA and device initiated. Partial requests can be supported. 1h: PSC_TRUE (default) 0h: PSC_FALSE
12:8	R	1Fh	T_AHCI_HBA_CAP_NCS: Number of Command Slots. This is a zero based value indicating the number of commands that can be cached in each of the ports implemented. Minimum number of command slots is 1 and maximum being 32. 0h: NCS_1 7h: NCS_8 Fh: NCS_16 1Fh: NCS_32 (default)
7	R	0	T_AHCI_HBA_CAP_CCCS: Command Completion Coalescing Support. If set to 1 it indicates that the controller supports command completion coalescing feature. If it is not set then T_AHCI_HBA_CCC_CTL and T_AHCI_HBA_CCC_PORTS are not implemented by the device. 1h: CCCS_TRUE 0h: CCCS_FALSE (default)
6	R	0	T_AHCI_HBA_CAP_EMS: Enclosure Management Support. If set to 1 it indicates the capability of the controller to support enclosure management. If it is not set then T_AHCI_HBA_EM_LOC and T_AHCI_HBA_EM_CTL are not implemented by the device. 1h: EMS_TRUE 0h: EMS_FALSE (default)
5	R	0	T_AHCI_HBA_CAP_SXS: Supports External SATA. This field indicates presence of 1 or many ports with the controller which support signal only connector for Serial ATA ports. The software should check the T_AHCI_PORT_PXCMD_ESP bit to find which port actually supports signal only connector. 1h: SXS_TRUE 0h: SXS_FALSE (default)
4:0	R	0	T_AHCI_HBA_CAP_NP: Number of ports. This is zero based value indicating maximum number of ports that is supported by the Host Bus Adapter. 1 port is the minimum requirement and a max of 32 (if not supporting command completion coalescing feature). 0h: NP_1 1h: NP_2 2h: NP_3 3h: NP_4 0h: NP_DEFAULT (default)

T_AHCI_HBA_GHC

HBA_GHC (Host Bus Adaptor) Global Host Control

This register indicates basic capabilities to Software.

Offset: 0x04 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31	R/W	0h	T_AHCI_HBA_GHC_AE: When set, indicates that communication to the HBA shall be via AHCI mechanisms. This can be used by an HBA that supports both legacy mechanisms (such as SFF-8038i) and AHCI to know when the HBA is running under an AHCI driver. When set, software shall only communicate with the HBA using AHCI. When cleared, software shall only communicate with the HBA using legacy mechanisms. When cleared FISes are not posted to memory and no commands are sent via AHCI mechanisms. Software shall set this bit to 1 before accessing other AHCI registers. The implementation of this bit is dependent upon the value of the CAP.SAM bit. If CAP.SAM is '0', then GHC.AE shall be read-write and shall have a reset value of '0'. If CAP.SAM is '1', then AE shall be read-only and shall have a reset value of '1'. 1h: AE_YES 0h: AE_NO (default)
30:3	R	0h	T_AHCI_HBA_GHC_RSVD: 0h: RSVD_VAL
2	R	0h	T_AHCI_HBA_GHC_MRSM: When set to 1 by hardware, indicates that the HBA requested more than one MSI vector but has reverted to using the first vector only. When this bit is cleared to 0, the HBA has not reverted to single MSI mode (i.e. hardware is already in single MSI mode, software has allocated the number of messages requested, or hardware is sharing interrupt vectors if MC.MME < MC.MMC). The HBA may revert to single MSI mode when the number of vectors allocated by the host is less than the number requested. This bit shall only be set to 1 when the following conditions hold: MC.MSIE = 1 (MSI is enabled) MC.MMC > 0 (multiple messages requested) MC.MME > 0 (more than one message allocated) MC.MME != MC.MMC (messages allocated not equal to number requested) When this bit is set to 1, single MSI mode operation is in use and software is responsible for clearing bits in the IS register to clear interrupts. This bit shall be cleared to 0 by hardware when any of the four conditions stated is false. This bit is also cleared to 0 when MC.MSIE = 1 and MC.MME = 0h. In this case, the hardware has been programmed to use single MSI mode, and is not reverting to that mode. 1h: MRSM_YES 0h: MRSM_NO (default)
1	R/W	0h	T_AHCI_HBA_GHC_IE: This global bit enables interrupts from the HBA. When cleared (reset default), all interrupt sources from all ports are disabled. When set, interrupts are enabled. 1h: IE_TRUE 0h: IE_FALSE (default)
0	R/W	0h	T_AHCI_HBA_GHC_HR: When set by SW, this bit causes an internal reset of the HBA. All state machines that relate to data transfers and queuing shall return to an idle condition, and all ports shall be re-initialized via COMRESET (if staggered spin-up is not supported). If staggered spin-up is supported, then it is the responsibility of software to spin-up each port after the reset has completed. When the HBA has performed the reset action, it shall reset this bit to 0. A software write of 0 shall have no effect. 0h: HR_NOT (default) 1h: HR_TRUE 1h: HR_SET

T_AHCI_HBA_IS

HBA_IS (Host Bus Adaptor) Interrupt Status

This register indicates basic capabilities to Software.

Offset: 0x08 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:4	R	0h	Reserved
3	RW1C	0h	T_AHCI_HBA_IS_PORT3_INTR: 0h: PORT3_INTR_ZERO (default) 1h: PORT3_INTR_SET 1h: PORT3_INTR_CLEAR
2	RW1C	0h	T_AHCI_HBA_IS_PORT2_INTR:

Bit	R/W	Reset	Description
			0h: PORT2_INTR_ZERO (default) 1h: PORT2_INTR_SET 1h: PORT2_INTR_CLEAR
1	RW1C	0h	T_AHCI_HBA_IS_PORT1_INTR: 0h: PORT1_INTR_ZERO (default) 1h: PORT1_INTR_SET 1h: PORT1_INTR_CLEAR
0	RW1C	0h	T_AHCI_HBA_IS_PORT0_INTR: 0h: PORT0_INTR_ZERO (default) 1h: PORT0_INTR_SET 1h: PORT0_INTR_CLEAR

T_AHCI_HBA_PI

HBA_PI (Host Bus Adaptor) Ports Implemented

This register indicates the ports exposed by the HBA. It is loaded by the BIOS.

Offset: 0x0c | Read/Write: R | Reset: 0x00000001

Bit	R/W	Reset	Description
31:0	R	1	T_AHCI_HBA_PI_PI: This register is bit significant. If a bit is set to 1, the corresponding port is available for software to use. If a bit is cleared to 0, the port is not available for software to use. The maximum number of bits set to 1 shall not exceed CAP.NP + 1, although the number of bits set in this register may be fewer than CAP.NP + 1. At least one bit shall be set to 1. 0h: PI_ZERO 1h: PI_FIRST 2h: PI_SECOND 4h: PI_THIRD 8h: PI_FOURTH 3Fh: PI_2PORTS_DIS Fh: PI_4PORTS_DIS 1h: PI_INIT (default)

T_AHCI_HBA_VS

HBA_VS AHCI Revision

This register indicates basic capabilities to software.

Offset: 0x10 | Read/Write: R | Reset: 0xXXXX0301

Bit	R/W	Reset	Description
31:16	R	1h	T_AHCI_HBA_VS_MAJOR_REV: Indicates the major version 1h: MAJOR_REV_1_3_1 1h: MAJOR_REV_1_3 1h: MAJOR_REV_1_2 1h: MAJOR_REV_1_0 0h: MAJOR_REV_0_95
15:0	R	0301h	T_AHCI_HBA_VS_MINOR_REV: Indicates the minor version NOTE: this field is reset by Cold Reset 301h: MINOR_REV_1_3_1 (default) 300h: MINOR_REV_1_3 200h: MINOR_REV_1_2 0h: MINOR_REV_1_0 905h: MINOR_REV_0_95

T_AHCI_HBA_CCC_CTL

HBA_CCC_CTL Command Completion Coalescing Control

This register indicates basic capabilities to Software.

Offset: 0x14 | Read/Write: R/W | Reset: 0xFFFFFFFx

Bit	R/W	Reset	Description
31:16	R/W	FFFFh	T_AHCI_HBA_CCC_CTL_TIMEOUT_VAL: The timeout value is specified in 1 millisecond intervals. The timer accuracy shall be within 5%. hCccTimer is loaded with this timeout value. hCccTimer is only decremented when commands are outstanding on selected ports. The HBA will signal a CCC interrupt when hCccTimer has decremented to 0. hCccTimer is reset to the timeout value on the assertion of each CCC interrupt. A timeout value of 0 is reserved. FFFFh: TIMEOUT_VAL_INIT (default)
15:8	R/W	FFh	T_AHCI_HBA_CCC_CTL_CC: Specifies the number of command completions that are necessary to cause a CCC interrupt. The HBA has an internal command completion counter, hCccComplete. hCccComplete is incremented by one each time a selected port has a command completion. When hCccComplete is equal to the command completions value, a CCC interrupt is signaled. The internal command completion counter is reset to 0 on the assertion of each CCC interrupt. A value of 0 for this field shall disable CCC interrupts being generated based on the number of commands completed, i.e. CCC interrupts are only generated based on the timer in this case. FFh: CC_INIT_VAL (default)
7:3	R	1Fh	T_AHCI_HBA_CCC_CTL_INT: Specifies the interrupt used by the CCC feature. This interrupt must be marked as unused in the Ports Implemented (PI) register by the corresponding bit being set to 0. Thus, the CCC interrupt corresponds to the interrupt for an unimplemented port on the controller. When a CCC interrupt occurs, the IS.IPS[INT] bit shall be asserted to 1. This field also specifies the interrupt vector used for MSI. 0h: INT_ZERO 4h: INT_FIVE 6h: INT_SEVEN 1Fh: INT_INIT (default)
2:1	R	0h	T_AHCI_HBA_CCC_CTL_RSVD: 0h: RSVD_ZEROS
0	R/W	0	T_AHCI_HBA_CCC_CTL_EN: When cleared to 0, the command completion coalescing feature is disabled and no CCC interrupts are generated. When set to 1, the command completion coalescing feature is enabled and CCC interrupts may be generated based on timeout or command completion conditions. Software shall only change the contents of the TV and CC fields when EN is cleared to 0. On transition of this bit from 0 to 1, any updated values for the TV and CC fields shall take effect. 1h: EN_SET 0h: EN_CLEAR (default)

T_AHCI_HBA_CCC_PORTS

HBA_CCC_PORTS Command Completion Coalescing Ports

This register indicates basic capabilities to Software.

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	T_AHCI_HBA_CCC_PORTS_PRT: When cleared to 0, the command completion coalescing feature is disabled and no CCC interrupts are generated. When set to 1, the command completion coalescing feature is enabled and CCC interrupts may be generated based on timeout or command completion conditions. Software shall only change the contents of the TV and CC fields when EN is cleared to 0. On transition of this bit from 0 to 1, any updated values for the TV and CC fields shall take effect. 0h: PRT_RESET_VAL (default)

T_AHCI_HBA_EM_LOC

HBA_EM_LOC Enclosure Management Location

This register indicates basic capabilities to Software.

Offset: 0x1c | Read/Write: R | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:16	R	0h	T_AHCI_HBA_EM_LOC_OFST: The offset of the message buffer in Dwords from the beginning of the ABAR. 0h: OFST_VAL
15:0	R	0h	T_AHCI_HBA_EM_LOC_SZ: Specifies the size of the transmit message buffer area in Dwords. If both transmit and receive buffers are supported, then the transmit buffer begins at ABAR[EM_LOC.OFST*4] and the receive buffer directly follows it. If both transmit and receive buffers are supported, both buffers are of the size indicated in the Buffer Size field. A value of 0 is invalid. 0h: SZ_VAL

T_AHCI_HBA_EM_CTL

HBA_EM_CTL Enclosure Management Control

This register indicates basic capabilities to Software.

Offset: 0x20 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:28	R	0h	T_AHCI_HBA_EM_CTL_RSVD3: 0h: RSVD3_VAL
27	R	0h	T_AHCI_HBA_EM_CTL_ATTR_PM: If set to 1, the HBA supports enclosure management messages for devices attached via a Port Multiplier. If cleared to 0, the HBA does not support enclosure management messages for devices attached via a Port Multiplier. When cleared to 0, software should use the Serial ATA enclosure management bridge that is built into many Port Multipliers for enclosure services with these devices. For more information on Serial ATA enclosure management bridges, refer to the Serial ATA Revision 2.6 specification. 0h: ATTR_PM_FALSE 1h: ATTR_PM_TRUE
26	R	0h	T_AHCI_HBA_EM_CTL_ATTR_ALHD: If set to 1, the HBA drives the activity LED for the LED message type in hardware and does not utilize software settings for this LED. The HBA does not begin transmitting the hardware based activity signal until after software has written CTL.TM=1 after a reset condition. 0h: ATTR_ALHD_FALSE 1h: ATTR_ALHD_TRUE
25	R	0h	T_AHCI_HBA_EM_CTL_ATTR_XMT: If set to 1, the HBA only supports transmitting messages and does not support receiving messages. If cleared to 0, the HBA supports transmitting and receiving messages. 0h: ATTR_XMT_FALSE 1h: ATTR_XMT_TRUE
24	R	0h	T_AHCI_HBA_EM_CTL_ATTR_SMB: If set to 1, the HBA has one message buffer that is shared for messages to transmit and messages received. In this case, unsolicited receive messages are not supported and it is softwares responsibility to manage access to this buffer. If cleared to 0, there are separate receive and transmit buffers such that unsolicited messages could be supported. 0h: ATTR_SMB_FALSE 1h: ATTR_SMB_TRUE
23:20	R	0h	T_AHCI_HBA_EM_CTL_RSVD2: 0h: RSVD2_VAL
19	R	0h	T_AHCI_HBA_EM_CTL_SUPP_SGPIO: If set to 1, the HBA supports the SGPIO register interface message type. 0h: SUPP_SGPIO_FALSE 1h: SUPP_SGPIO_TRUE
18	R	0h	T_AHCI_HBA_EM_CTL_SUPP_SES2: If set to 1, the HBA supports the SES-2 message type. 0h: SUPP_SES2_FALSE 1h: SUPP_SES2_TRUE
17	R	0h	T_AHCI_HBA_EM_CTL_SUPP_SAFTE: If set to 1, the HBA supports the SAF-TE message type.

Bit	R/W	Reset	Description
			0h: SUPP_SAFTE_FALSE 1h: SUPP_SAFTE_TRUE
16	R	0h	T_AHCI_HBA_EM_CTL_SUPP_LED: If set to 1, the HBA supports the LED message 0h: SUPP_LED_FALSE 1h: SUPP_LED_TRUE
15:10	R	0h	T_AHCI_HBA_EM_CTL_RSVD1: 0h: RSVD1_VAL
9	RW1C	0h	T_AHCI_HBA_EM_CTL_RST: When set to 1 by software, the HBA shall reset all enclosure management message logic and the attached enclosure processor (if applicable) and take all appropriate reset actions to ensure messages can be transmitted/received after the reset. After the HBA completes the reset operation, the HBA shall set the value to 0. A write of 0 by software to this field shall have no effect. 1h: RST_SET 0h: RST_CLEAR (default)
8	RW1C	0h	T_AHCI_HBA_EM_CTL_TM: When set to 1 by software, the HBA shall transmit the message contained in the message buffer. When the message is completely sent, the HBA shall clear this bit to 0. A write of 0 to this bit by software shall have no effect. Software shall not change the contents of the message buffer while CTL.TM is set to 1. 0h: TM_CLEAR (default) 1h: TM_SET
7:1	R	0h	T_AHCI_HBA_EM_CTL_RSVD0: 0h: RSVD0_VAL
0	RW1C	0h	T_AHCI_HBA_EM_CTL_STS_MR: The HBA sets this bit to a 1 when a message is completely received into the message buffer. When software detects this bit is a 1, software should read the message and perform any actions necessary. When software is finished reading the message in the buffer, software writes a 1 to this bit in order to clear it. A write of 0 to this bit by software shall have no effect. 1h: STS_MR_TRUE 0h: STS_MR_FALSE (default) 1h: STS_MR_CLEAR

T_AHCI_HBA_CAP2

CAP2 - HBA Capabilities Extended

This register indicates basic capabilities to Software.

Offset: 0x24 | Read/Write: R | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:6	R	0h	T_AHCI_HBA_CAP2_RSVD_1: 0h: RSVD_1_VAL
5	R	1	T_AHCI_HBA_CAP2_DESO: This field specifies that the HBA shall only assert the DEVSLP signal if the interface is in Slumber. When this bit is set to 1, the HBA shall ignore software directed entrance to DevSleep via PxCMD.ICC unless PxSSTS.IPM = 6h. When this bit is cleared to 0, the HBA may enter DevSleep from any link state (active, Partial, or Slumber). NOTE: this field is reset by Cold Reset 1h: DESO_TRUE (default) 0h: DESO_FALSE
4	R	1	T_AHCI_HBA_CAP2_SADM: When set to 1, the HBA supports hardware assertion of the DEVSLP signal after the idle timeout expires. When cleared to 0, this function is not supported and software shall treat the PxDEVSLP.ADSE field as reserved. NOTE: this field is reset by Cold Reset 1h: SADM_TRUE (default) 0h: SADM_FALSE
3	R	1	T_AHCI_HBA_CAP2_SDS: When set to 1, the HBA supports the Device Sleep feature. When cleared to 0, DEVSLP is not supported and software shall not set PxCMD.ICC to '8h' NOTE: this field is reset by Cold Reset 1h: SDS_TRUE (default)

Bit	R/W	Reset	Description
			0h: SDS_FALSE
2	R	1	T_AHCI_HBA_CAP2_APST: When set to 1, the HBA supports Automatic Partial to Slumber Transitions. When cleared to 0, Automatic Partial to Slumber Transitions are not supported. NOTE: this field is reset by Cold Reset 1h: APST_TRUE (default) 0h: APST_FALSE
1	R	0h	T_AHCI_HBA_CAP2_RSVD_0: 0h: RSVD_0_VAL
0	R	0h	T_AHCI_HBA_CAP2_BOH: When set to 1, the HBA supports the BIOS/OS handoff mechanism. When cleared to 0, the HBA does not support the BIOS/OS handoff mechanism. When BIOS/OS handoff is supported, the HBA has implemented the BOHC global HBA register. When cleared to 0, it indicates that the HBA does not support BIOS/OS handoff and the BOHC global HBA register is not implemented. NOTE: this field is reset by Cold Reset 1h: BOH_TRUE 0h: BOH_FALSE (default)

T_AHCI_HBA_BOHC

BOHC - BIOS/OS Handoff Control and Status

This register controls various global actions of the HBA

Offset: 0x28 | Read/Write: R | Reset: 0xXXXXXXX0

Bit	R/W	Reset	Description
31:5	R	0h	T_AHCI_HBA_BOHC_RSVD: 0h: RSVD_VAL
4	R/W	0h	T_AHCI_HBA_BOHC_BB: BIOS Busy (BB) 1h: BB_TRUE 0h: BB_FALSE (default)
3	RW1C	0h	T_AHCI_HBA_BOHC_OOC: OS Ownership Change (OOC) 1h: OOC_TRUE 0h: OOC_FALSE (default) 1h: OOC_CLEAR
2	R/W	0h	T_AHCI_HBA_BOHC_SOOE: SMI on OS Ownership Change Enable (SOOE) 1h: SOOE_TRUE 0h: SOOE_FALSE (default)
1	R/W	0h	T_AHCI_HBA_BOHC_OOS: OS Owned Semaphore (OOS) 1h: OOS_TRUE 0h: OOS_FALSE (default)
0	R/W	0h	T_AHCI_HBA_BOHC_BOS: BIOS Owned Semaphore (BOS) 1h: BOS_TRUE 0h: BOS_FALSE (default)

T_AHCI_HBA_CAP_BKDR

Offset: 0xa0 | Read/Write: R/W | Reset: 0xE620FF01

Bit	Reset	R/W	Description
31	1	R/W	T_AHCI_HBA_CAP_BKDR_S64A: 1h: S64A_TRUE (default) 0h: S64A_FALSE
30	1	R/W	T_AHCI_HBA_CAP_BKDR_SNCQ: 1h: SNCQ_TRUE (default) 0h: SNCQ_FALSE

Bit	Reset	R/W	Description
29	1	R/W	T_AHCI_HBA_CAP_BKDR_SSNTF: 1h: SSNTF_TRUE (default) 0h: SSNTF_FALSE
28	0	R/W	T_AHCI_HBA_CAP_BKDR_SMPS: 1h: SMPS_TRUE 0h: SMPS_FALSE (default)
27	0	R/W	T_AHCI_HBA_CAP_BKDR_SUPP_STG_SPUP: Backdoor field to advertise staggered spin up. BIOS has to write this we are going to support staggered spin up. 1h: SUPP_STG_SPUP_SET 0h: SUPP_STG_SPUP_CLEAR (default)
26	1	R/W	T_AHCI_HBA_CAP_BKDR_SALP: 1h: SALP_TRUE (default) 0h: SALP_FALSE
25	1	R/W	T_AHCI_HBA_CAP_BKDR_SAL: 1h: SAL_TRUE (default) 0h: SAL_FALSE
24	0	R/W	T_AHCI_HBA_CAP_BKDR_SUPP_CLO: 1h: SUPP_CLO_TRUE 0h: SUPP_CLO_FALSE (default)
23:20	2h	R/W	T_AHCI_HBA_CAP_BKDR_INTF_SPD_SUPP: CAP.ISS can be written using this 0h: INTF_SPD_SUPP_RSVD 1h: INTF_SPD_SUPP_GEN1 2h: INTF_SPD_SUPP_GEN1_2 (default)
19	0	R/W	T_AHCI_HBA_CAP_BKDR_SUPP_NONZERO_OFFSET: 1h: SUPP_NONZERO_OFFSET_TRUE 0h: SUPP_NONZERO_OFFSET_FALSE (default)
18	0	R/W	T_AHCI_HBA_CAP_BKDR_SUPP_AHCI_ONLY: 1h: SUPP_AHCI_ONLY_TRUE 0h: SUPP_AHCI_ONLY_FALSE (default)
17	0	R/W	T_AHCI_HBA_CAP_BKDR_SUPP_PM: 1h: SUPP_PM_TRUE 0h: SUPP_PM_FALSE (default)
16	0	R/W	T_AHCI_HBA_CAP_BKDR_FIS_SWITCHING: 1h: FIS_SWITCHING_TRUE 0h: FIS_SWITCHING_FALSE (default)
15	1	R/W	T_AHCI_HBA_CAP_BKDR_PIO_MULT_DRQ_BLK: 1h: PIO_MULT_DRQ_BLK_SUPP (default) 0h: PIO_MULT_DRQ_BLK_NOT_SUPP
14	1	R/W	T_AHCI_HBA_CAP_BKDR_SLUMBER_ST_CAP: 1h: SLUMBER_ST_CAP_TRUE (default) 0h: SLUMBER_ST_CAP_FALSE
13	1	R/W	T_AHCI_HBA_CAP_BKDR_PARTIAL_ST_CAP: 1h: PARTIAL_ST_CAP_TRUE (default) 0h: PARTIAL_ST_CAP_FALSE
12:8	1Fh	R/W	T_AHCI_HBA_CAP_BKDR_NUM_CMD_SLOTS: 0h: NUM_CMD_SLOTS_1 7h: NUM_CMD_SLOTS_8 Fh: NUM_CMD_SLOTS_16 1Fh: NUM_CMD_SLOTS_32 (default)
7	0	R/W	T_AHCI_HBA_CAP_BKDR_CMD_CMPL_COALESING: 1h: CMD_CMPL_COALESING_TRUE 0h: CMD_CMPL_COALESING_FALSE (default)
6	0	R/W	T_AHCI_HBA_CAP_BKDR_ENCL_MGMT_SUPP: 1h: ENCL_MGMT_SUPP_TRUE 0h: ENCL_MGMT_SUPP_FALSE (default)

Bit	Reset	R/W	Description
5	0	R/W	T_AHCI_HBA_CAP_BKDR_EXT_SATA: This is a backdoor access for enabling external SATA. 1h: EXT_SATA_SUPPORTED 0h: EXT_SATA_NOT_SUPPORTED (default)
4:0	1	R/W	T_AHCI_HBA_CAP_BKDR_NUM_PORTS: CAP.NP can be written using this 0h: NUM_PORTS_1 1h: NUM_PORTS_2 2h: NUM_PORTS_3 3h: NUM_PORTS_4 1h: NUM_PORTS_DEFUALT (default)

T_AHCI_HBA_SPARE

Spare register with cold reset

Offset: 0xa4 | Read/Write: R/W | Reset: 0x7FFFFFFF

Bit	R/W	Reset	Description
31:0	R/W	7FFFFFFFh	T_AHCI_HBA_SPARE_0_RSVD: 7FFFFFFFh: RSVD_INIT (default)

T_AHCI_HBA_PLL_CTRL

Offset: 0xa8 | Read/Write: R/W | Reset: 0x00000008

Bit	R/W	Reset	Description
31:25	R	0	Reserved
24	R/W	0	T_AHCI_HBA_PLL_CTRL_CLAMP_TXCLK_ON_DEVSLP: NOTE: this field is reset by Cold Reset 1h: CLAMP_TXCLK_ON_DEVSLP_YES 0h: CLAMP_TXCLK_ON_DEVSLP_NO (default)
23	R/W	0	T_AHCI_HBA_PLL_CTRL_CLAMP_DEVCLK_ON_DEVSLP: NOTE: this field is reset by Cold Reset 1h: CLAMP_DEVCLK_ON_DEVSLP_YES 0h: CLAMP_DEVCLK_ON_DEVSLP_NO (default)
22	R/W	0	T_AHCI_HBA_PLL_CTRL_SHUTDOWN_TXCLK_ON_DEVSLP: NOTE: this field is reset by Cold Reset 1h: SHUTDOWN_TXCLK_ON_DEVSLP_YES 0h: SHUTDOWN_TXCLK_ON_DEVSLP_NO (default)
21	R/W	0	T_AHCI_HBA_PLL_CTRL_DIS_TXRXCLK_SHUTDOWN: NOTE: this field is reset by Cold Reset 1h: DIS_TXRXCLK_SHUTDOWN_YES 0h: DIS_TXRXCLK_SHUTDOWN_NO (default)
20	R/W	0	T_AHCI_HBA_PLL_CTRL_DIS_TXRXCLK_CLAMPING: NOTE: this field is reset by Cold Reset 1h: DIS_TXRXCLK_CLAMPING_YES 0h: DIS_TXRXCLK_CLAMPING_NO (default)
19	R/W	0	T_AHCI_HBA_PLL_CTRL_DIS_DEVCLK_CLAMPING: NOTE: this field is reset by Cold Reset 1h: DIS_DEVCLK_CLAMPING_YES 0h: DIS_DEVCLK_CLAMPING_NO (default)
18	R/W	0	T_AHCI_HBA_PLL_CTRL_CLAMP_DEVCLK_IN_AWAIT_COMINIT: NOTE: this field is reset by Cold Reset 1h: CLAMP_DEVCLK_IN_AWAIT_COMINIT_YES 0h: CLAMP_DEVCLK_IN_AWAIT_COMINIT_NO (default)
17	R/W	0	T_AHCI_HBA_PLL_CTRL_CLAMP_TXRXCLK_IN_AWAIT_COMINIT: NOTE: this field is reset by Cold Reset 1h: CLAMP_TXRXCLK_IN_AWAIT_COMINIT_YES

Bit	R/W	Reset	Description
			0h: CLAMP_TXRXCLK_IN_AWAIT_COMINIT_NO (default)
16	R/W	0	T_AHCI_HBA_PLL_CTRL_UNCLAMP_DEVCLK_ONLY_THRU_SW: NOTE: this field is reset by Cold Reset 1h: UNCLAMP_DEVCLK_ONLY_THRU_SW_YES 0h: UNCLAMP_DEVCLK_ONLY_THRU_SW_NO (default)
15	R/W	0	T_AHCI_HBA_PLL_CTRL_UNCLAMP_TXRXCLK_ONLY_THRU_SW: NOTE: this field is reset by Cold Reset 1h: UNCLAMP_TXRXCLK_ONLY_THRU_SW_YES 0h: UNCLAMP_TXRXCLK_ONLY_THRU_SW_NO (default)
14	R/W	0	T_AHCI_HBA_PLL_CTRL_WAKEUP_ONLY_THRU_SW: NOTE: this field is reset by Cold Reset 1h: WAKEUP_ONLY_THRU_SW_YES 0h: WAKEUP_ONLY_THRU_SW_NO (default)
13	R/W	0	T_AHCI_HBA_PLL_CTRL_CLAMP_TXCLK_ON_SLUMBER: This bit set indicates that txclk and rxclk clamping is enabled when PHY is in SLUMBER and the SHUTDOWN signal would not be asserted NOTE: this field is reset by Cold Reset 1h: CLAMP_TXCLK_ON_SLUMBER_YES 0h: CLAMP_TXCLK_ON_SLUMBER_NO (default)
12	R/W	0	T_AHCI_HBA_PLL_CTRL_CLAMP_TXCLK_ON_PHYRDY_LOW: This bit set indicates txclk and rxclk clamping is enabled when PHY_RDY goes low NOTE: this field is reset by Cold Reset 1h: CLAMP_TXCLK_ON_PHYRDY_LOW_YES 0h: CLAMP_TXCLK_ON_PHYRDY_LOW_NO (default)
11	R/W	0	T_AHCI_HBA_PLL_CTRL_CLAMP_TXCLK_ON_PARTIAL: This bit set indicates that txclk and rxclk clamping is enabled when PHY is in PARTIAL NOTE: this field is reset by Cold Reset 1h: CLAMP_TXCLK_ON_PARTIAL_YES 0h: CLAMP_TXCLK_ON_PARTIAL_NO (default)
10	R/W	0	T_AHCI_HBA_PLL_CTRL_CLAMP_DEVCLK_ON_SLUMBER: NOTE: this field is reset by Cold Reset 1h: CLAMP_DEVCLK_ON_SLUMBER_YES 0h: CLAMP_DEVCLK_ON_SLUMBER_NO (default)
9	R/W	0	T_AHCI_HBA_PLL_CTRL_CLAMP_DEVCLK_ON_PHYRDY_LOW: NOTE: this field is reset by Cold Reset 1h: CLAMP_DEVCLK_ON_PHYRDY_LOW_YES 0h: CLAMP_DEVCLK_ON_PHYRDY_LOW_NO (default)
8	R/W	0	T_AHCI_HBA_PLL_CTRL_CLAMP_DEVCLK_ON_PARTIAL: NOTE: this field is reset by Cold Reset 1h: CLAMP_DEVCLK_ON_PARTIAL_YES 0h: CLAMP_DEVCLK_ON_PARTIAL_NO (default)
7	R/W	0	T_AHCI_HBA_PLL_CTRL_SHUTDOWN_TXCLK_ON_PHYRDY_LOW: When this bit is one and all ports PHY_RDY goes low, shutdown signal will be asserted. Similar is the case for the DEVCLK clamp and shutdown bits NOTE: this field is reset by Cold Reset 1h: SHUTDOWN_TXCLK_ON_PHYRDY_LOW_YES 0h: SHUTDOWN_TXCLK_ON_PHYRDY_LOW_NO (default)
6	R/W	0	T_AHCI_HBA_PLL_CTRL_SHUTDOWN_TXCLK_ON_SLUMBER: When the above bit is zero and this bit is one the SHUTDOWN signal would be asserted when all the ports are in SLUMBER NOTE: this field is reset by Cold Reset 1h: SHUTDOWN_TXCLK_ON_SLUMBER_YES 0h: SHUTDOWN_TXCLK_ON_SLUMBER_NO (default)
5	R/W	0	T_AHCI_HBA_PLL_CTRL_SHUTDOWN_DEVCLK_ON_PHYRDY_LOW: NOTE: this field is reset by Cold Reset 1h: SHUTDOWN_DEVCLK_ON_PHYRDY_LOW_YES 0h: SHUTDOWN_DEVCLK_ON_PHYRDY_LOW_NO (default)
4	R/W	0	T_AHCI_HBA_PLL_CTRL_SHUTDOWN_DEVCLK_ON_SLUMBER: NOTE: this field is reset by Cold Reset

Bit	R/W	Reset	Description
			1h: SHUTDOWN_DEVCLK_ON_SLUMBER_YES 0h: SHUTDOWN_DEVCLK_ON_SLUMBER_NO (default)
3	R/W	1	T_AHCI_HBA_PLL_CTRL_NO_CLAMP_SHUTDOWN: When this bit is set the clamping of clocks(all clocks) and shutdown of PLL is disabled. It should be written to zero if CLAMP/SHUTDOWN needs to be enabled NOTE: this field is reset by Cold Reset 1h: NO_CLAMP_SHUTDOWN_YES (default) 0h: NO_CLAMP_SHUTDOWN_NO
2	R/W	0h	T_AHCI_HBA_PLL_CTRL_TXRXCLK_WAKEUP: Deasserts the above signal to wakeup the TXRXCLK PLL 0h: TXRXCLK_WAKEUP_NO (default) 1h: TXRXCLK_WAKEUP_YES 1h: TXRXCLK_WAKEUP_SET
1	R/W	0h	T_AHCI_HBA_PLL_CTRL_TXRXCLK_SHUTDOWN: Sets a signal to clocks block to shut down the TXRXCLK PLL 0h: TXRXCLK_SHUTDOWN_NO (default) 1h: TXRXCLK_SHUTDOWN_YES 1h: TXRXCLK_SHUTDOWN_SET
0	R/W	0h	T_AHCI_HBA_PLL_CTRL_DEVCLK_SHUTDOWN: Sets a signal to clocks block to shut down the DEVCLK PLL 0h: DEVCLK_SHUTDOWN_NO (default) 1h: DEVCLK_SHUTDOWN_YES 1h: DEVCLK_SHUTDOWN_SET

T_AHCI_HBA_SHUTDOWN_TIMER

Offset: 0xac | Read/Write: R/W | Reset: 0x00010004

Bit	R/W	Reset	Description
31:28	R	0	Reserved
27:14	R/W	0004h	T_AHCI_HBA_SHUTDOWN_TIMER_DEV_CLK: 4h: DEV_CLK_INIT (default)
13:0	R/W	0004h	T_AHCI_HBA_SHUTDOWN_TIMER_TXRX_CLK: Specifies how many 100us to wait before shutting down the PLL 4h: TXRX_CLK_INIT (default)

T_AHCI_HBA_SPARE_3

Offset: 0xf4 | Read/Write: R/W | Reset: 0xF0F0F0F0

Bit	R/W	Reset	Description
31:16	R/W	F0F0h	T_AHCI_HBA_SPARE_3_RSVD_31_16: NOTE: this field is reset by Cold Reset F0F0h: RSVD_31_16_ZERO (default)
15:0	R/W	F0F0h	T_AHCI_HBA_SPARE_3_RSVD_15_00: F0F0h: RSVD_15_00_ZERO (default)

31.7.4.2 Port Registers

T_AHCI_PORT_PXCLB

Port Command List Base Address

Offset: 0x100 | Read/Write: R/W | Reset: 0x00000XXX

Bit	R/W	Reset	Description

Bit	R/W	Reset	Description
31:10	R/W	0h	T_AHCI_PORT_PXCLB_CLB: Indicates the 32-bit base physical address for the command list for this port. This base is used when fetching commands to execute. The structure pointed to by this address range is 1K-bytes in length. This address must be 1K-byte aligned as indicated by bits 09:00 being read only. 0h: CLB_00 (default)
9:0	R	0h	T_AHCI_PORT_PXCLB_RSVD: 0h: RSVD_00

T_AHCI_PORT_PXCLBU

Port Command List Base Address Upper

Offset: 0x104 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	T_AHCI_PORT_PXCLBU_CLB: Indicates the upper 32 bits for the command list base physical address for this port. This base is used when fetching commands to execute. This register shall be read only 0 for HBAs that do not support 64-bit addressing 0h: CLB_00 (default)

T_AHCI_PORT_PXFB

Port FIS Base Address

Offset: 0x108 | Read/Write: R/W | Reset: 0x000000XX

Bit	R/W	Reset	Description
31:8	R/W	0h	T_AHCI_PORT_PXFB_FB: Indicates the 32-bit base physical address for received FISes. The structure pointed to by this address range is 256 bytes in length. This address must be 256-byte aligned as indicated by bits 07:00 being read only. When FIS-based switching is in use, this structure is 4KB in length and the address shall be 4KB aligned 0h: FB_00 (default)
7:0	R	0h	T_AHCI_PORT_PXFB_RSVD: 0h: RSVD_00

T_AHCI_PORT_PXFBU

Port FIS Base Address Upper

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

Bit	R/W	Reset	Description
31:0	R/W	0	T_AHCI_PORT_PXFBU_FB: Indicates the upper 32 bits for the received FIS base physical address for this port. This register shall be read only 0 for HBAs that do not support 64-bit addressing. 0h: FB_00 (default)

T_AHCI_PORT_PXIS

Port Interrupt Status

Offset: 0x110 | Read/Write: R/W | Reset: 0x00XXXXX0

Bit	R/W	Reset	Description
31	RW1C	0h	T_AHCI_PORT_PXIS_CPDS: 0h: CPDS_NOTRCVD (default) 1h: CPDS_RCVD 1h: CPDS_CLEAR

Bit	R/W	Reset	Description
30	RW1C	0h	T_AHCI_PORT_PXIS_TFES: 0h: TFES_NOTRCVD (default) 1h: TFES_RCVD 1h: TFES_CLEAR
29	RW1C	0h	T_AHCI_PORT_PXIS_HBFS: 0h: HBFS_NOTRCVD (default) 1h: HBFS_RCVD 1h: HBFS_CLEAR
28	RW1C	0h	T_AHCI_PORT_PXIS_HBDS: 0h: HBDS_NOTRCVD (default) 1h: HBDS_RCVD 1h: HBDS_CLEAR
27	RW1C	0h	T_AHCI_PORT_PXIS_IFS: 0h: IFS_NOTRCVD (default) 1h: IFS_RCVD 1h: IFS_CLEAR
26	RW1C	0h	T_AHCI_PORT_PXIS_INFS: 0h: INFS_NOTRCVD (default) 1h: INFS_RCVD 1h: INFS_CLEAR
25	R	0h	Reserved
24	RW1C	0h	T_AHCI_PORT_PXIS_OFS: 0h: OFS_NOTRCVD (default) 1h: OFS_RCVD 1h: OFS_CLEAR
23	RW1C	0h	T_AHCI_PORT_PXIS_IPMS: 0h: IPMS_NOTRCVD (default) 1h: IPMS_RCVD 1h: IPMS_CLEAR
22	R	0h	T_AHCI_PORT_PXIS_PRCS: 0h: PRCS_CLEAR (default) 1h: PRCS_SET
21:8	R	0h	T_AHCI_PORT_PXIS_RSVD: 0h: RSVD_00
7	R	0h	T_AHCI_PORT_PXIS_DMPS: 0h: DMPS_OPEN 1h: DMPS_CLOSED 1h: DMPS_CLEAR
6	R	0h	T_AHCI_PORT_PXIS_PCS: 1h: PCS_TRUE 0h: PCS_FALSE (default)
5	RW1C	0h	T_AHCI_PORT_PXIS_DPS: 0h: DPS_NOTDONE (default) 1h: DPS_DONE 1h: DPS_CLEAR
4	R	0h	T_AHCI_PORT_PXIS_UFS: 0h: UFS_NOTPRSNT (default) 1h: UFS_PRSNT
3	RW1C	0h	T_AHCI_PORT_PXIS_SDBS: 0h: SDBS_NOTPRSNT (default) 1h: SDBS_PRSNT 1h: SDBS_CLEAR
2	RW1C	0h	T_AHCI_PORT_PXIS_DSS: 0h: DSS_NOTPRSNT (default) 1h: DSS_PRSNT 1h: DSS_CLEAR

Bit	R/W	Reset	Description
1	RW1C	0h	T_AHCI_PORT_PXIS_PSS: 0h: PSS_NOTPRSNT (default) 1h: PSS_PRSNT 1h: PSS_CLEAR
0	RW1C	0h	T_AHCI_PORT_PXIS_DHRS: 0h: DHRS_NOTPRSNT (default) 1h: DHRS_PRSNT 1h: DHRS_CLEAR

T_AHCI_PORT_PXIE

Port Interrupt Enable

Offset: 0x114 | Read/Write: R/W | Reset: 0x00XXXXX0

Bit	R/W	Reset	Description
31	R	0h	T_AHCI_PORT_PXIE_CPDE: 0h: CPDE_CLEAR (default) 1h: CPDE_SET
30	R/W	0h	T_AHCI_PORT_PXIE_TFEE: 0h: TFEE_CLEAR (default) 1h: TFEE_SET
29	R/W	0h	T_AHCI_PORT_PXIE_HBFE: 0h: HBFE_CLEAR (default) 1h: HBFE_SET
28	R/W	0h	T_AHCI_PORT_PXIE_HBDE: 0h: HBDE_CLEAR (default) 1h: HBDE_SET
27	R/W	0h	T_AHCI_PORT_PXIE_IFE: 0h: IFE_CLEAR (default) 1h: IFE_SET
26	R/W	0h	T_AHCI_PORT_PXIE_INFE: 0h: INFE_CLEAR (default) 1h: INFE_SET
25	R	0h	Reserved
24	R/W	0h	T_AHCI_PORT_PXIE_OFE: 0h: OFE_CLEAR (default) 1h: OFE_SET
23	R/W	0h	T_AHCI_PORT_PXIE_IPME: 0h: IPME_CLEAR (default) 1h: IPME_SET
22	R/W	0h	T_AHCI_PORT_PXIE_PRCE: 0h: PRCE_CLEAR (default) 1h: PRCE_SET
21:8	R	0h	T_AHCI_PORT_PXIE_RSVD: 0h: RSVD_00
7	R	0h	T_AHCI_PORT_PXIE_DMPE: 0h: DMPE_CLEAR 1h: DMPE_SET
6	R/W	0h	T_AHCI_PORT_PXIE_PCE: 0h: PCE_CLEAR (default) 1h: PCE_SET
5	R/W	0h	T_AHCI_PORT_PXIE_DPE: 0h: DPE_CLEAR (default) 1h: DPE_SET

Bit	R/W	Reset	Description
4	R/W	0h	T_AHCI_PORT_PXIE_UFE: 0h: UFE_CLEAR (default) 1h: UFE_SET
3	R/W	0h	T_AHCI_PORT_PXIE_SDBE: 0h: SDBE_CLEAR (default) 1h: SDBE_SET
2	R/W	0h	T_AHCI_PORT_PXIE_DSE: 1h: DSE_SET 0h: DSE_CLEAR (default)
1	R/W	0h	T_AHCI_PORT_PXIE_PSE: 0h: PSE_CLEAR (default) 1h: PSE_SET
0	R/W	0h	T_AHCI_PORT_PXIE_DHRE: 0h: DHRE_CLEAR (default) 1h: DHRE_SET

T_AHCI_PORT_PXCMD

Port Command and Status

Offset: 0x118 | Read/Write: R/W | Reset: 0x000X0004

Bit	R/W	Reset	Description
31:28	R/W	0h	T_AHCI_PORT_PXCMD_ICC: 0h: ICC_IDLE (default) 0h: ICC_NO_OP 1h: ICC_ACTIVE 2h: ICC_PARTIAL 6h: ICC_SLUMBER 8h: ICC_DEVSLEEP
27	R/W	0h	T_AHCI_PORT_PXCMD_ASP: 0h: ASP_DISABLE (default) 1h: ASP_ENABLE
26	R/W	0h	T_AHCI_PORT_PXCMD_ALPE: 1h: ALPE_TRUE 0h: ALPE_FALSE (default)
25	R/W	0h	T_AHCI_PORT_PXCMD_DLAE: 1h: DLAE_DRV_LED_ALWYS 0h: DLAE_DRV_LED_CMDS (default)
24	R/W	0h	T_AHCI_PORT_PXCMD_ATAPI: 0h: ATAPI_DEVICE_NOTPRSNT (default) 1h: ATAPI_DEVICE_PRSNT
23	R/W	0h	T_AHCI_PORT_PXCMD_APSTE: 1h: APSTE_TRUE 0h: APSTE_FALSE (default)
22	R	0h	T_AHCI_PORT_PXCMD_FBSCP: 1h: FBSCP_TRUE 0h: FBSCP_FALSE (default)
21	R	0h	T_AHCI_PORT_PXCMD_ESP: 0h: ESP_NOTSUPPORTED (default) 1h: ESP_SUPPORTED
20	R	0h	T_AHCI_PORT_PXCMD_CPD: 0h: CPD_NOTSUPPORTED (default) 1h: CPD_SUPPORTED
19	R	0h	T_AHCI_PORT_PXCMD_MPSP: 0h: MPSP_FALSE

Bit	R/W	Reset	Description
			1h: MPSP_TRUE
18	R	1	T_AHCI_PORT_PXCMD_HPCP: 0h: HPCP_FALSE 1h: HPCP_TRUE (default)
17	R/W	0h	T_AHCI_PORT_PXCMD_PMA: 0h: PMA_FALSE (default) 1h: PMA_TRUE
16	R	0h	T_AHCI_PORT_PXCMD_CPS: 0h: CPS_NOTDETECT (default) 1h: CPS_DETECT
15	R	0h	T_AHCI_PORT_PXCMD_CR: 0h: CR_NOTRCVD (default) 1h: CR_RCVD
14	R	0h	T_AHCI_PORT_PXCMD_FR: 0h: FR_NOTRCVD (default) 1h: FR_RCVD
13	R	0h	T_AHCI_PORT_PXCMD_MPSS: 0h: MPSS_CLOSED (default) 1h: MPSS_OPEN
12:8	R	0h	T_AHCI_PORT_PXCMD_CCS: 0h: CCS_00 (default) Fh: CCS_16 1Fh: CCS_32
7:5	R	0h	Reserved
4	R/W	0h	T_AHCI_PORT_PXCMD_FRE: 0h: FRE_CLEAR (default) 1h: FRE_SET
3	R/W	0h	T_AHCI_PORT_PXCMD_CLO: 0h: CLO_DISABLE (default) 1h: CLO_ENABLE
2	R	1h	T_AHCI_PORT_PXCMD_POD: 0h: POD_CLEAR 1h: POD_SET (default)
1	R/W	0h	T_AHCI_PORT_PXCMD_SUD: 0h: SUD_CLEAR (default) 1h: SUD_SET
0	R/W	0h	T_AHCI_PORT_PXCMD_ST: 0h: ST_CLEAR (default) 1h: ST_SET

T_AHCI_PORT_PXTFD

Port Task File Data

Offset: 0x120 | Read/Write: R/W | Reset: 0xXXXX007F

Bit	R/W	Reset	Description
31:16	R	0h	T_AHCI_PORT_PXTFD_RSVD: 0h: RSVD_00
15:8	R	0h	T_AHCI_PORT_PXTFD_ERR: 0h: ERR_00 (default)
7	R	0h	T_AHCI_PORT_PXTFD_STS_BSY: 0h: STS_BSY_CLEAR (default) 1h: STS_BSY_SET

Bit	R/W	Reset	Description
6	R	1h	T_AHCI_PORT_PXTFD_STS_DRDY: 0h: STS_DRDY_CLEAR 1h: STS_DRDY_SET (default)
5	R	1h	T_AHCI_PORT_PXTFD_STS_DF: 0h: STS_DF_CLEAR 1h: STS_DF_SET (default)
4	R	1h	T_AHCI_PORT_PXTFD_STS_CS: 0h: STS_CS_CLEAR 1h: STS_CS_SET (default)
3	R	1h	T_AHCI_PORT_PXTFD_STS_DRQ: 0h: STS_DRQ_CLEAR 1h: STS_DRQ_SET (default)
2:1	R	3h	T_AHCI_PORT_PXTFD_STS_2_1: 3h: STS_2_1_INIT (default)
0	R	1h	T_AHCI_PORT_PXTFD_STS_ERR: 0h: STS_ERR_CLEAR 1h: STS_ERR_SET (default)

T_AHCI_PORT_PXSIG

Port Signature

Offset: 0x124 | Read/Write: R/W | Reset: 0xFFFFFFFF

Bit	R/W	Reset	Description
31:24	R	FFh	T_AHCI_PORT_PXSIG_LBA_HIGH: FFh: LBA_HIGH_INIT (default) 0h: LBA_HIGH_00
23:16	R	FFh	T_AHCI_PORT_PXSIG_LBA_MID: FFh: LBA_MID_INIT (default) 0h: LBA_MID_00
15:8	R	FFh	T_AHCI_PORT_PXSIG_LBA_LOW: FFh: LBA_LOW_INIT (default) 0h: LBA_LOW_00
7:0	R	FFh	T_AHCI_PORT_PXSIG_SECTOR_CNT: FFh: SECTOR_CNT_INIT (default) 0h: SECTOR_CNT_00

T_AHCI_PORT_PXSSTS

Offset: 0x128 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:12	R	0h	Reserved
11:8	R	0h	T_AHCI_PORT_PXSSTS_IPM: 0h: IPM_NO_DEV (default) 1h: IPM_ACT_ST 2h: IPM_PARTIAL_ST 6h: IPM_SLUMBER_ST 8h: IPM_DEVSLEEP_ST
7:4	R	0h	T_AHCI_PORT_PXSSTS_SPD: 0h: SPD_NO_DEV (default) 1h: SPD_GEN1 2h: SPD_GEN2
3:0	R	0h	T_AHCI_PORT_PXSSTS_DET:

Bit	R/W	Reset	Description
			0h: DET_NO_DEV (default) 1h: DET_DEV_PRSNT_NO_PHY_COMM 3h: DET_DEV_PRSNT_PHY_COMM 4h: DET_BIST_LPBK

T_AHCI_PORT_PXSCTL

Port Serial ATA Control

Offset: 0x12c | Read/Write: R/W | Reset: 0x000XX000

Bit	R/W	Reset	Description
31:20	R	0h	Reserved
19:16	R	0h	T_AHCI_PORT_PXSCTL_PMP: 0h: PMP_00
15:12	R	0h	T_AHCI_PORT_PXSCTL_SPM: 0h: SPM_00
11:8	R/W	0h	T_AHCI_PORT_PXSCTL_IPM: 0h: IPM_NO_RSTCT (default) 1h: IPM_PARTIAL_ST_DISABLED 2h: IPM_SLUMBER_ST_DISABLED 3h: IPM_PART_SLUM_ST_DISABLED 4h: IPM_DEVSLEEP_ST_DISABLED 5h: IPM_PART_DEVSLP_ST_DISABLED 6h: IPM_SLUM_DEVSLP_ST_DISABLED 7h: IPM_PART_SLUM_DEVSLP_ST_DISABLED
7:4	R/W	0h	T_AHCI_PORT_PXSCTL_SPD: 0h: SPD_00 (default) 1h: SPD_GEN1 2h: SPD_GEN2
3:0	R/W	0h	T_AHCI_PORT_PXSCTL_DET: 0h: DET_NO_DEV_RQD (default) 1h: DET_INTF_INIT 4h: DET_PHY_OFFLINE

T_AHCI_PORT_PXSERR

Port Serial ATA Error

Offset: 0x130 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:27	R	0h	Reserved
26	RW1C	0h	T_AHCI_PORT_PXSERR_DIAG_X: 0h: DIAG_X_FALSE (default) 1h: DIAG_X_TRUE 1h: DIAG_X_CLEAR
25	RW1C	0h	T_AHCI_PORT_PXSERR_DIAG_F: 0h: DIAG_F_FALSE (default) 1h: DIAG_F_TRUE 1h: DIAG_F_CLEAR
24	RW1C	0h	T_AHCI_PORT_PXSERR_DIAG_T: 0h: DIAG_T_FALSE (default) 1h: DIAG_T_TRUE 1h: DIAG_T_CLEAR
23	RW1C	0h	T_AHCI_PORT_PXSERR_DIAG_S: 0h: DIAG_S_FALSE (default)

Bit	R/W	Reset	Description
			1h: DIAG_S_TRUE 1h: DIAG_S_CLEAR
22	RW1C	0h	T_AHCI_PORT_PXSERR_DIAG_H: 0h: DIAG_H_FALSE (default) 1h: DIAG_H_TRUE 1h: DIAG_H_CLEAR
21	RW1C	0h	T_AHCI_PORT_PXSERR_DIAG_C: 0h: DIAG_C_FALSE (default) 1h: DIAG_C_TRUE 1h: DIAG_C_CLEAR
20	RW1C	0h	T_AHCI_PORT_PXSERR_DIAG_D: 0h: DIAG_D_FALSE (default) 1h: DIAG_D_TRUE 1h: DIAG_D_CLEAR
19	RW1C	0h	T_AHCI_PORT_PXSERR_DIAG_B: 0h: DIAG_B_FALSE (default) 1h: DIAG_B_TRUE 1h: DIAG_B_CLEAR
18	RW1C	0h	T_AHCI_PORT_PXSERR_DIAG_W: 0h: DIAG_W_FALSE (default) 1h: DIAG_W_TRUE 1h: DIAG_W_CLEAR
17	RW1C	0h	T_AHCI_PORT_PXSERR_DIAG_I: 0h: DIAG_I_FALSE (default) 1h: DIAG_I_TRUE 1h: DIAG_I_CLEAR
16	RW1C	0h	T_AHCI_PORT_PXSERR_DIAG_N: 0h: DIAG_N_FALSE (default) 1h: DIAG_N_TRUE 1h: DIAG_N_CLEAR
15:12	R	0h	Reserved
11	RW1C	0h	T_AHCI_PORT_PXSERR_ERR_E: 0h: ERR_E_FALSE (default) 1h: ERR_E_TRUE 1h: ERR_E_CLEAR
10	RW1C	0h	T_AHCI_PORT_PXSERR_ERR_P: 0h: ERR_P_FALSE (default) 1h: ERR_P_TRUE 1h: ERR_P_CLEAR
9	RW1C	0h	T_AHCI_PORT_PXSERR_ERR_C: 0h: ERR_C_FALSE (default) 1h: ERR_C_TRUE 1h: ERR_C_CLEAR
8	RW1C	0h	T_AHCI_PORT_PXSERR_ERR_T: 0h: ERR_T_FALSE (default) 1h: ERR_T_TRUE 1h: ERR_T_CLEAR
7:2	R	0h	Reserved
1	RW1C	0h	T_AHCI_PORT_PXSERR_ERR_M: 0h: ERR_M_FALSE (default) 1h: ERR_M_TRUE 1h: ERR_M_CLEAR
0	RW1C	0h	T_AHCI_PORT_PXSERR_ERR_I: 0h: ERR_I_FALSE (default) 1h: ERR_I_TRUE 1h: ERR_I_CLEAR

T_AHCI_PORT_PXSACT

Port Serial ATA Active

Offset: 0x134 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31	R/W	0h	T_AHCI_PORT_PXSACT_DS31: 0h: DS31_NOT_PRSNT (default) 1h: DS31_SET 1h: DS31_PRSNT
30	R/W	0h	T_AHCI_PORT_PXSACT_DS30: 0h: DS30_NOT_PRSNT (default) 1h: DS30_SET 1h: DS30_PRSNT
29	R/W	0h	T_AHCI_PORT_PXSACT_DS29: 0h: DS29_NOT_PRSNT (default) 1h: DS29_SET 1h: DS29_PRSNT
28	R/W	0h	T_AHCI_PORT_PXSACT_DS28: 0h: DS28_NOT_PRSNT (default) 1h: DS28_SET 1h: DS28_PRSNT
27	R/W	0h	T_AHCI_PORT_PXSACT_DS27: 0h: DS27_NOT_PRSNT (default) 1h: DS27_SET 1h: DS27_PRSNT
26	R/W	0h	T_AHCI_PORT_PXSACT_DS26: 0h: DS26_NOT_PRSNT (default) 1h: DS26_SET 1h: DS26_PRSNT
25	R/W	0h	T_AHCI_PORT_PXSACT_DS25: 0h: DS25_NOT_PRSNT (default) 1h: DS25_SET 1h: DS25_PRSNT
24	R/W	0h	T_AHCI_PORT_PXSACT_DS24: 0h: DS24_NOT_PRSNT (default) 1h: DS24_SET 1h: DS24_PRSNT
23	R/W	0h	T_AHCI_PORT_PXSACT_DS23: 0h: DS23_NOT_PRSNT (default) 1h: DS23_SET 1h: DS23_PRSNT
22	R/W	0h	T_AHCI_PORT_PXSACT_DS22: 0h: DS22_NOT_PRSNT (default) 1h: DS22_SET 1h: DS22_PRSNT
21	R/W	0h	T_AHCI_PORT_PXSACT_DS21: 0h: DS21_NOT_PRSNT (default) 1h: DS21_SET 1h: DS21_PRSNT
20	R/W	0h	T_AHCI_PORT_PXSACT_DS20: 0h: DS20_NOT_PRSNT (default) 1h: DS20_SET 1h: DS20_PRSNT
19	R/W	0h	T_AHCI_PORT_PXSACT_DS19: 0h: DS19_NOT_PRSNT (default) 1h: DS19_SET 1h: DS19_PRSNT

Bit	R/W	Reset	Description
18	R/W	0h	T_AHCI_PORT_PXSACT_DS18: 0h: DS18_NOT_PRSNT (default) 1h: DS18_SET 1h: DS18_PRSNT
17	R/W	0h	T_AHCI_PORT_PXSACT_DS17: 0h: DS17_NOT_PRSNT (default) 1h: DS17_SET 1h: DS17_PRSNT
16	R/W	0h	T_AHCI_PORT_PXSACT_DS16: 0h: DS16_NOT_PRSNT (default) 1h: DS16_SET 1h: DS16_PRSNT
15	R/W	0h	T_AHCI_PORT_PXSACT_DS15: 0h: DS15_NOT_PRSNT (default) 1h: DS15_SET 1h: DS15_PRSNT
14	R/W	0h	T_AHCI_PORT_PXSACT_DS14: 0h: DS14_NOT_PRSNT (default) 1h: DS14_SET 1h: DS14_PRSNT
13	R/W	0h	T_AHCI_PORT_PXSACT_DS13: 0h: DS13_NOT_PRSNT (default) 1h: DS13_SET 1h: DS13_PRSNT
12	R/W	0h	T_AHCI_PORT_PXSACT_DS12: 0h: DS12_NOT_PRSNT (default) 1h: DS12_SET 1h: DS12_PRSNT
11	R/W	0h	T_AHCI_PORT_PXSACT_DS11: 0h: DS11_NOT_PRSNT (default) 1h: DS11_SET 1h: DS11_PRSNT
10	R/W	0h	T_AHCI_PORT_PXSACT_DS10: 0h: DS10_NOT_PRSNT (default) 1h: DS10_SET 1h: DS10_PRSNT
9	R/W	0h	T_AHCI_PORT_PXSACT_DS9: 0h: DS9_NOT_PRSNT (default) 1h: DS9_SET 1h: DS9_PRSNT
8	R/W	0h	T_AHCI_PORT_PXSACT_DS8: 0h: DS8_NOT_PRSNT (default) 1h: DS8_SET 1h: DS8_PRSNT
7	R/W	0h	T_AHCI_PORT_PXSACT_DS7: 0h: DS7_NOT_PRSNT (default) 1h: DS7_SET 1h: DS7_PRSNT
6	R/W	0h	T_AHCI_PORT_PXSACT_DS6: 0h: DS6_NOT_PRSNT (default) 1h: DS6_SET 1h: DS6_PRSNT
5	R/W	0h	T_AHCI_PORT_PXSACT_DS5: 0h: DS5_NOT_PRSNT (default) 1h: DS5_SET 1h: DS5_PRSNT
4	R/W	0h	T_AHCI_PORT_PXSACT_DS4:

Bit	R/W	Reset	Description
			0h: DS4_NOT_PRSNT (default) 1h: DS4_SET 1h: DS4_PRSNT
3	R/W	0h	T_AHCI_PORT_PXSACT_DS3: 0h: DS3_NOT_PRSNT (default) 1h: DS3_SET 1h: DS3_PRSNT
2	R/W	0h	T_AHCI_PORT_PXSACT_DS2: 0h: DS2_NOT_PRSNT (default) 1h: DS2_SET 1h: DS2_PRSNT
1	R/W	0h	T_AHCI_PORT_PXSACT_DS1: 0h: DS1_NOT_PRSNT (default) 1h: DS1_SET 1h: DS1_PRSNT
0	R/W	0h	T_AHCI_PORT_PXSACT_DS0: 0h: DS0_NOT_PRSNT (default) 1h: DS0_SET 1h: DS0_PRSNT

T_AHCI_PORT_PXCI

Port Command Issue

Offset: 0x138 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31	R/W	0h	T_AHCI_PORT_PXCI_CI31: 0h: CI31_CMD_NOT_PRSNT (default) 1h: CI31_CMD_PRSNT 1h: CI31_CMD_SET
30	R/W	0h	T_AHCI_PORT_PXCI_CI30: 0h: CI30_CMD_NOT_PRSNT (default) 1h: CI30_CMD_PRSNT 1h: CI30_CMD_SET
29	R/W	0h	T_AHCI_PORT_PXCI_CI29: 0h: CI29_CMD_NOT_PRSNT (default) 1h: CI29_CMD_PRSNT 1h: CI29_CMD_SET
28	R/W	0h	T_AHCI_PORT_PXCI_CI28: 0h: CI28_CMD_NOT_PRSNT (default) 1h: CI28_CMD_PRSNT 1h: CI28_CMD_SET
27	R/W	0h	T_AHCI_PORT_PXCI_CI27: 0h: CI27_CMD_NOT_PRSNT (default) 1h: CI27_CMD_PRSNT 1h: CI27_CMD_SET
26	R/W	0h	T_AHCI_PORT_PXCI_CI26: 0h: CI26_CMD_NOT_PRSNT (default) 1h: CI26_CMD_PRSNT 1h: CI26_CMD_SET
25	R/W	0h	T_AHCI_PORT_PXCI_CI25: 0h: CI25_CMD_NOT_PRSNT (default) 1h: CI25_CMD_PRSNT 1h: CI25_CMD_SET
24	R/W	0h	T_AHCI_PORT_PXCI_CI24: 0h: CI24_CMD_NOT_PRSNT (default) 1h: CI24_CMD_PRSNT

Bit	R/W	Reset	Description
			1h: CI24_CMD_SET
23	R/W	0h	T_AHCI_PORT_PXCI_CI23: 0h: CI23_CMD_NOT_PRSNT (default) 1h: CI23_CMD_PRSNT 1h: CI23_CMD_SET
22	R/W	0h	T_AHCI_PORT_PXCI_CI22: 0h: CI22_CMD_NOT_PRSNT (default) 1h: CI22_CMD_PRSNT 1h: CI22_CMD_SET
21	R/W	0h	T_AHCI_PORT_PXCI_CI21: 0h: CI21_CMD_NOT_PRSNT (default) 1h: CI21_CMD_PRSNT 1h: CI21_CMD_SET
20	R/W	0h	T_AHCI_PORT_PXCI_CI20: 0h: CI20_CMD_NOT_PRSNT (default) 1h: CI20_CMD_PRSNT 1h: CI20_CMD_SET
19	R/W	0h	T_AHCI_PORT_PXCI_CI19: 0h: CI19_CMD_NOT_PRSNT (default) 1h: CI19_CMD_PRSNT 1h: CI19_CMD_SET
18	R/W	0h	T_AHCI_PORT_PXCI_CI18: 0h: CI18_CMD_NOT_PRSNT (default) 1h: CI18_CMD_PRSNT 1h: CI18_CMD_SET
17	R/W	0h	T_AHCI_PORT_PXCI_CI17: 0h: CI17_CMD_NOT_PRSNT (default) 1h: CI17_CMD_PRSNT 1h: CI17_CMD_SET
16	R/W	0h	T_AHCI_PORT_PXCI_CI16: 0h: CI16_CMD_NOT_PRSNT (default) 1h: CI16_CMD_PRSNT 1h: CI16_CMD_SET
15	R/W	0h	T_AHCI_PORT_PXCI_CI15: 0h: CI15_CMD_NOT_PRSNT (default) 1h: CI15_CMD_PRSNT 1h: CI15_CMD_SET
14	R/W	0h	T_AHCI_PORT_PXCI_CI14: 0h: CI14_CMD_NOT_PRSNT (default) 1h: CI14_CMD_PRSNT 1h: CI14_CMD_SET
13	R/W	0h	T_AHCI_PORT_PXCI_CI13: 0h: CI13_CMD_NOT_PRSNT (default) 1h: CI13_CMD_PRSNT 1h: CI13_CMD_SET
12	R/W	0h	T_AHCI_PORT_PXCI_CI12: 0h: CI12_CMD_NOT_PRSNT (default) 1h: CI12_CMD_PRSNT 1h: CI12_CMD_SET
11	R/W	0h	T_AHCI_PORT_PXCI_CI11: 0h: CI11_CMD_NOT_PRSNT (default) 1h: CI11_CMD_PRSNT 1h: CI11_CMD_SET
10	R/W	0h	T_AHCI_PORT_PXCI_CI10: 0h: CI10_CMD_NOT_PRSNT (default) 1h: CI10_CMD_PRSNT 1h: CI10_CMD_SET

Bit	R/W	Reset	Description
9	R/W	0h	T_AHCI_PORT_PXCI_CI9: 0h: CI9_CMD_NOT_PRSNT (default) 1h: CI9_CMD_PRSNT 1h: CI9_CMD_SET
8	R/W	0h	T_AHCI_PORT_PXCI_CI8: 0h: CI8_CMD_NOT_PRSNT (default) 1h: CI8_CMD_PRSNT 1h: CI8_CMD_SET
7	R/W	0h	T_AHCI_PORT_PXCI_CI7: 0h: CI7_CMD_NOT_PRSNT (default) 1h: CI7_CMD_PRSNT 1h: CI7_CMD_SET
6	R/W	0h	T_AHCI_PORT_PXCI_CI6: 0h: CI6_CMD_NOT_PRSNT (default) 1h: CI6_CMD_PRSNT 1h: CI6_CMD_SET
5	R/W	0h	T_AHCI_PORT_PXCI_CI5: 0h: CI5_CMD_NOT_PRSNT (default) 1h: CI5_CMD_PRSNT 1h: CI5_CMD_SET
4	R/W	0h	T_AHCI_PORT_PXCI_CI4: 0h: CI4_CMD_NOT_PRSNT (default) 1h: CI4_CMD_PRSNT 1h: CI4_CMD_SET
3	R/W	0h	T_AHCI_PORT_PXCI_CI3: 0h: CI3_CMD_NOT_PRSNT (default) 1h: CI3_CMD_PRSNT 1h: CI3_CMD_SET
2	R/W	0h	T_AHCI_PORT_PXCI_CI2: 0h: CI2_CMD_NOT_PRSNT (default) 1h: CI2_CMD_PRSNT 1h: CI2_CMD_SET
1	R/W	0h	T_AHCI_PORT_PXCI_CI1: 0h: CI1_CMD_NOT_PRSNT (default) 1h: CI1_CMD_PRSNT 1h: CI1_CMD_SET
0	R/W	0h	T_AHCI_PORT_PXCI_CI0: 0h: CI0_CMD_NOT_PRSNT (default) 1h: CI0_CMD_PRSNT 1h: CI0_CMD_SET

T_AHCI_PORT_PXSNTF

Port S Notification

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

Bit	R/W	Reset	Description
31:16	R	0h	Reserved
15	RW1C	0h	T_AHCI_PORT_PXSNTF_PMN_PORT15: 0h: PMN_PORT15_NOT_SET (default) 1h: PMN_PORT15_SET 1h: PMN_PORT15_CLEAR
14	RW1C	0h	T_AHCI_PORT_PXSNTF_PMN_PORT14: 0h: PMN_PORT14_NOT_SET (default) 1h: PMN_PORT14_SET 1h: PMN_PORT14_CLEAR

Bit	R/W	Reset	Description
13	RW1C	0h	T_AHCI_PORT_PXSNTF_PMN_PORT13: 0h: PMN_PORT13_NOT_SET (default) 1h: PMN_PORT13_SET 1h: PMN_PORT13_CLEAR
12	RW1C	0h	T_AHCI_PORT_PXSNTF_PMN_PORT12: 0h: PMN_PORT12_NOT_SET (default) 1h: PMN_PORT12_SET 1h: PMN_PORT12_CLEAR
11	RW1C	0h	T_AHCI_PORT_PXSNTF_PMN_PORT11: 0h: PMN_PORT11_NOT_SET (default) 1h: PMN_PORT11_SET 1h: PMN_PORT11_CLEAR
10	RW1C	0h	T_AHCI_PORT_PXSNTF_PMN_PORT10: 0h: PMN_PORT10_NOT_SET (default) 1h: PMN_PORT10_SET 1h: PMN_PORT10_CLEAR
9	RW1C	0h	T_AHCI_PORT_PXSNTF_PMN_PORT9: 0h: PMN_PORT9_NOT_SET (default) 1h: PMN_PORT9_SET 1h: PMN_PORT9_CLEAR
8	RW1C	0h	T_AHCI_PORT_PXSNTF_PMN_PORT8: 0h: PMN_PORT8_NOT_SET (default) 1h: PMN_PORT8_SET 1h: PMN_PORT8_CLEAR
7	RW1C	0h	T_AHCI_PORT_PXSNTF_PMN_PORT7: 0h: PMN_PORT7_NOT_SET (default) 1h: PMN_PORT7_SET 1h: PMN_PORT7_CLEAR
6	RW1C	0h	T_AHCI_PORT_PXSNTF_PMN_PORT6: 0h: PMN_PORT6_NOT_SET (default) 1h: PMN_PORT6_SET 1h: PMN_PORT6_CLEAR
5	RW1C	0h	T_AHCI_PORT_PXSNTF_PMN_PORT5: 0h: PMN_PORT5_NOT_SET (default) 1h: PMN_PORT5_SET 1h: PMN_PORT5_CLEAR
4	RW1C	0h	T_AHCI_PORT_PXSNTF_PMN_PORT4: 0h: PMN_PORT4_NOT_SET (default) 1h: PMN_PORT4_SET 1h: PMN_PORT4_CLEAR
3	RW1C	0h	T_AHCI_PORT_PXSNTF_PMN_PORT3: 0h: PMN_PORT3_NOT_SET (default) 1h: PMN_PORT3_SET 1h: PMN_PORT3_CLEAR
2	RW1C	0h	T_AHCI_PORT_PXSNTF_PMN_PORT2: 0h: PMN_PORT2_NOT_SET (default) 1h: PMN_PORT2_SET 1h: PMN_PORT2_CLEAR
1	RW1C	0h	T_AHCI_PORT_PXSNTF_PMN_PORT1: 0h: PMN_PORT1_NOT_SET (default) 1h: PMN_PORT1_SET 1h: PMN_PORT1_CLEAR
0	RW1C	0h	T_AHCI_PORT_PXSNTF_PMN_PORT0: 0h: PMN_PORT0_NOT_SET (default) 1h: PMN_PORT0_SET 1h: PMN_PORT0_CLEAR

T_AHCI_PORT_PXFBS

Reserved for FIS Based Switiching

Offset: 0x140 | Read/Write: R/W | Reset: 0xXXX0F0Xx

Bit	R/W	Reset	Description
31:20	R	0h	T_AHCI_PORT_PXFBS_RSVD_31_20: 0h: RSVD_31_20_0
19:16	R	0h	T_AHCI_PORT_PXFBS_DWE: 0h: DWE_DEFAULT (default)
15:12	R	Fh	T_AHCI_PORT_PXFBS_ADO: Fh: ADO_DEFAULT (default)
11:8	R/W	0h	T_AHCI_PORT_PXFBS_DEV: 0h: DEV_DEFAULT (default)
7:3	R	0h	T_AHCI_PORT_PXFBS_RSVD_7_3: 0h: RSVD_7_3_0
2	R	0h	T_AHCI_PORT_PXFBS_SDE: 0h: SDE_DEFAULT (default)
1	R/W	0h	T_AHCI_PORT_PXFBS_DEC: 0h: DEC_NOT (default) 1h: DEC_TRUE 1h: DEC_SET
0	R	0h	T_AHCI_PORT_PXFBS_EN: 0h: EN_FALSE

T_AHCI_PORT_PXDEVSLP

Port Device Sleep

Offset: 0x144 | Read/Write: R/W | Reset: 0xX0052450

Bit	R/W	Reset	Description
31:29	R	0h	T_AHCI_PORT_PXDEVSLP_RSVD_31_29: 0h: RSVD_31_29_0
28:25	R	0h	T_AHCI_PORT_PXDEVSLP_DM: 0h: DM_DEFAULT (default)
24:15	R/W	00Ah	T_AHCI_PORT_PXDEVSLP_DITO: Ah: DITO_DEFAULT (default)
14:10	R/W	09h	T_AHCI_PORT_PXDEVSLP_MDAT: 9h: MDAT_DEFAULT (default)
9:2	R/W	14h	T_AHCI_PORT_PXDEVSLP_DETO: 14h: DETO_DEFAULT (default)
1	R	0h	T_AHCI_PORT_PXDEVSLP_DSP: 0h: DSP_TRUE (default) 1h: DSP_FALSE
0	R/W	0h	T_AHCI_PORT_PXDEVSLP_ADSE: 0h: ADSE_TRUE (default) 1h: ADSE_FALSE

T_AHCI_PORT_BKDR

Port Vendor Specific Register

Offset: 0x170 | Read/Write: R/W | Reset: 0x00000100

Bit	R/W	Reset	Description
31:24	R/W	0h	T_AHCI_PORT_BKDR_PXDEVSLP_DETO_OVERRIDE_VAL: 0h: PXDEVSLP_DETO_OVERRIDE_VAL_DEFAULT (default)
23:21	R	0h	Reserved
20:16	R/W	0h	T_AHCI_PORT_BKDR_PXDEVSLP_MDAT_OVERRIDE_VAL: 0h: PXDEVSLP_MDAT_OVERRIDE_VAL_DEFAULT (default)
15	R/W	0h	T_AHCI_PORT_BKDR_PXDEVSLP_DETO_OVERRIDE: 0h: PXDEVSLP_DETO_OVERRIDE_DEFAULT (default)
14	R/W	0h	T_AHCI_PORT_BKDR_PXDEVSLP_MDAT_OVERRIDE: 0h: PXDEVSLP_MDAT_OVERRIDE_DEFAULT (default)
13:10	R/W	0h	T_AHCI_PORT_BKDR_PXDEVSLP_DM: 0h: PXDEVSLP_DM_DEFAULT (default)
9	R/W	0h	T_AHCI_PORT_BKDR_PORT_UNCONNECTED: This is to be written by the BIOS during the initial boot-up when it finds a port unconnected NOTE: this field is reset by Cold Reset 1h: PORT_UNCONNECTED_YES 0h: PORT_UNCONNECTED_NO (default) 1h: PORT_UNCONNECTED_DONE
8	R/W	1	T_AHCI_PORT_BKDR_CLK_CLAMP_CTRL_CLAMP_THIS_CH: This is used to enable/disable Clamping capability for individual channel(both devclk and txclk) NO_CLAMP_SHUTDOWN should be made 0 if CLAMP_THIS_CH are used. NOTE: this field is reset by Cold Reset 1h: CLK_CLAMP_CTRL_CLAMP_THIS_CH_YES (default) 0h: CLK_CLAMP_CTRL_CLAMP_THIS_CH_NO
7	R/W	0h	T_AHCI_PORT_BKDR_CLK_CLAMP_CTRL_TXRXCLK_UNCLAMP: 0h: CLK_CLAMP_CTRL_TXRXCLK_UNCLAMP_NO (default) 1h: CLK_CLAMP_CTRL_TXRXCLK_UNCLAMP_YES 1h: CLK_CLAMP_CTRL_TXRXCLK_UNCLAMP_SET
6	R/W	0h	T_AHCI_PORT_BKDR_CLK_CLAMP_CTRL_TXRXCLK_CLAMP: 0h: CLK_CLAMP_CTRL_TXRXCLK_CLAMP_NO (default) 1h: CLK_CLAMP_CTRL_TXRXCLK_CLAMP_YES 1h: CLK_CLAMP_CTRL_TXRXCLK_CLAMP_SET
5	R/W	0h	T_AHCI_PORT_BKDR_CLK_CLAMP_CTRL_DEVCLK_UNCLAMP: 0h: CLK_CLAMP_CTRL_DEVCLK_UNCLAMP_NO (default) 1h: CLK_CLAMP_CTRL_DEVCLK_UNCLAMP_YES 1h: CLK_CLAMP_CTRL_DEVCLK_UNCLAMP_SET
4	R/W	0h	T_AHCI_PORT_BKDR_CLK_CLAMP_CTRL_DEVCLK_CLAMP: 0h: CLK_CLAMP_CTRL_DEVCLK_CLAMP_NO (default) 1h: CLK_CLAMP_CTRL_DEVCLK_CLAMP_YES 1h: CLK_CLAMP_CTRL_DEVCLK_CLAMP_SET
3	R/W	0h	T_AHCI_PORT_BKDR_HOTPLUG_CAP: NOTE: this field is reset by Cold Reset 1h: HOTPLUG_CAP_SUPP 0h: HOTPLUG_CAP_NOT_SUPP (default)
2	R/W	0h	T_AHCI_PORT_BKDR_MECH_SWITCH: NOTE: this field is reset by Cold Reset 1h: MECH_SWITCH_SUPP 0h: MECH_SWITCH_NOT_SUPP (default)
1	R/W	0h	T_AHCI_PORT_BKDR_COLD_PRSN_DET: NOTE: this field is reset by Cold Reset 1h: COLD_PRSN_DET_SUPP 0h: COLD_PRSN_DET_NOT_SUPP (default)
0	R/W	0h	T_AHCI_PORT_BKDR_EXT_SATA_SUPP: NOTE: this field is reset by Cold Reset 1h: EXT_SATA_SUPP_TRUE 0h: EXT_SATA_SUPP_FALSE (default)

T_AHCI_PORT_INTR

Port Interrupt Source

Each of the field indicates what specific condition caused the DUT to raise a fatal/non-fatal interrupt. Each of the fields can be masked by the corresponding mask register.

Offset: 0x174 | Read/Write: R/W | Reset: 0xFFFB0000

Bit	R/W	Reset	Description
31	R/W	1	T_AHCI_PORT_INTR_EN_IFS_NCQ_NON_NCQ_MIX: 0h: EN_IFS_NCQ_NON_NCQ_MIX_CLEARED 1h: EN_IFS_NCQ_NON_NCQ_MIX_SET (default)
30	R/W	1	T_AHCI_PORT_INTR_EN_IFS_FPDMA_TAG_ERROR: 0h: EN_IFS_FPDMA_TAG_ERROR_CLEARED 1h: EN_IFS_FPDMA_TAG_ERROR_SET (default)
29	R/W	1	T_AHCI_PORT_INTR_EN_INFS_NONDATA_FIS_CRC_ERROR: 0h: EN_INFS_NONDATA_FIS_CRC_ERROR_CLEARED 1h: EN_INFS_NONDATA_FIS_CRC_ERROR_SET (default)
28	R/W	1	T_AHCI_PORT_INTR_EN_INFS_PMP_MISMATCH: 0h: EN_INFS_PMP_MISMATCH_CLEARED 1h: EN_INFS_PMP_MISMATCH_SET (default)
27	R/W	1	T_AHCI_PORT_INTR_EN_INFS_NONDATA_FIS_SIZE: 0h: EN_INFS_NONDATA_FIS_SIZE_CLEARED 1h: EN_INFS_NONDATA_FIS_SIZE_SET (default)
26	R/W	1	T_AHCI_PORT_INTR_EN_INFS_UFIS_SIZE: 0h: EN_INFS_UFIS_SIZE_CLEARED 1h: EN_INFS_UFIS_SIZE_SET (default)
25	R/W	1	T_AHCI_PORT_INTR_EN_IFS_DMA_CNT_ERROR: 0h: EN_IFS_DMA_CNT_ERROR_CLEARED 1h: EN_IFS_DMA_CNT_ERROR_SET (default)
24	R/W	1	T_AHCI_PORT_INTR_EN_IFS_FIFO_OVERWRITTEN: 0h: EN_IFS_FIFO_OVERWRITTEN_CLEARED 1h: EN_IFS_FIFO_OVERWRITTEN_SET (default)
23	R/W	1	T_AHCI_PORT_INTR_EN_IFS_SYNC_ESCAPE_RECV: 0h: EN_IFS_SYNC_ESCAPE_RECV_CLEARED 1h: EN_IFS_SYNC_ESCAPE_RECV_SET (default)
22	R/W	1	T_AHCI_PORT_INTR_EN_IFS_NONNCQ_TX_RX_UNDERFLOW: 0h: EN_IFS_NONNCQ_TX_RX_UNDERFLOW_CLEARED 1h: EN_IFS_NONNCQ_TX_RX_UNDERFLOW_SET (default)
21	R/W	1	T_AHCI_PORT_INTR_EN_IFS_DATA_FIS_RECV: 0h: EN_IFS_DATA_FIS_RECV_CLEARED 1h: EN_IFS_DATA_FIS_RECV_SET (default)
20	R/W	1	T_AHCI_PORT_INTR_EN_IFS_DATA_FIS_XMIT: 0h: EN_IFS_DATA_FIS_XMIT_CLEARED 1h: EN_IFS_DATA_FIS_XMIT_SET (default)
19	R/W	1	T_AHCI_PORT_INTR_EN_IFS_ATAPI_CMD_XMIT: 0h: EN_IFS_ATAPI_CMD_XMIT_CLEARED 1h: EN_IFS_ATAPI_CMD_XMIT_SET (default)
18	R/W	0	T_AHCI_PORT_INTR_EN_IFS_NCQ_TX_RX_UNDERFLOW: 0h: EN_IFS_NCQ_TX_RX_UNDERFLOW_CLEARED 0h: EN_IFS_NCQ_TX_RX_UNDERFLOW_SET (default)
17	R/W	1	T_AHCI_PORT_INTR_EN_OFS_TX: 0h: EN_OFS_TX_CLEARED 1h: EN_OFS_TX_SET (default)
16	R/W	1	T_AHCI_PORT_INTR_EN_OFS_RX: 0h: EN_OFS_RX_CLEARED

Bit	R/W	Reset	Description
			1h: EN_OFS_RX_SET (default)
15	R	0	T_AHCI_PORT_INTR_IFS_NCQ_NON_NCQ_MIX: 0h: IFS_NCQ_NON_NCQ_MIX_INIT (default) 1h: IFS_NCQ_NON_NCQ_MIX_SET
14	R	0	T_AHCI_PORT_INTR_IFS_FPDMA_TAG_ERROR: 0h: IFS_FPDMA_TAG_ERROR_CLEAR (default) 1h: IFS_FPDMA_TAG_ERROR_SET
13	R	0	T_AHCI_PORT_INTR_INFS_NONDATA_FIS_CRC_ERROR: 0h: INFS_NONDATA_FIS_CRC_ERROR_CLEAR (default) 1h: INFS_NONDATA_FIS_CRC_ERROR_SET
12	R	0	T_AHCI_PORT_INTR_INFS_PMP_MISMATCH: 0h: INFS_PMP_MISMATCH_CLEAR (default) 1h: INFS_PMP_MISMATCH_SET
11	R	0	T_AHCI_PORT_INTR_INFS_NONDATA_FIS_SIZE: 0h: INFS_NONDATA_FIS_SIZE_CLEAR (default) 1h: INFS_NONDATA_FIS_SIZE_SET
10	R	0	T_AHCI_PORT_INTR_INFS_UFIS_SIZE: 0h: INFS_UFIS_SIZE_CLEAR (default) 1h: INFS_UFIS_SIZE_SET
9	R	0	T_AHCI_PORT_INTR_IFS_DMA_CNT_ERROR: 0h: IFS_DMA_CNT_ERROR_CLEAR (default) 1h: IFS_DMA_CNT_ERROR_SET
8	R	0	T_AHCI_PORT_INTR_IFS_FIFO_OVERWRITTEN: 0h: IFS_FIFO_OVERWRITTEN_CLEAR (default) 1h: IFS_FIFO_OVERWRITTEN_SET
7	R	0	T_AHCI_PORT_INTR_IFS_SYNC_ESCAPE_RECV: 0h: IFS_SYNC_ESCAPE_RECV_CLEAR (default) 1h: IFS_SYNC_ESCAPE_RECV_SET
6	R	0	T_AHCI_PORT_INTR_IFS_NONNCQ_TX_RX: 0h: IFS_NONNCQ_TX_RX_CLEAR (default) 1h: IFS_NONNCQ_TX_RX_SET
5	R	0	T_AHCI_PORT_INTR_IFS_DATA_FIS_RECV: 0h: IFS_DATA_FIS_RECV_CLEAR (default) 1h: IFS_DATA_FIS_RECV_SET
4	R	0	T_AHCI_PORT_INTR_IFS_DATA_FIS_XMIT: 0h: IFS_DATA_FIS_XMIT_CLEAR (default) 1h: IFS_DATA_FIS_XMIT_SET
3	R	0	T_AHCI_PORT_INTR_IFS_ATAPI_CMD_XMIT: 0h: IFS_ATAPI_CMD_XMIT_CLEAR (default) 1h: IFS_ATAPI_CMD_XMIT_SET
2	R	0	T_AHCI_PORT_INTR_IFS_NCQ_TX_RX: 0h: IFS_NCQ_TX_RX_CLEAR (default) 1h: IFS_NCQ_TX_RX_SET
1	R	0	T_AHCI_PORT_INTR_OFS_TX: 0h: OFS_TX_CLEAR (default) 1h: OFS_TX_SET
0	R	0	T_AHCI_PORT_INTR_OFS_RX: 0h: OFS_RX_CLEAR (default) 1h: OFS_RX_SET

T_AHCI_PORT_FBS_RX_PMP_REG_FIS

This register implements the per PMP status of register FIS reception

Offset: 0x178 | Read/Write: R/W | Reset: 0xXXXX0000

Bit	R/W	Reset	Description
31:16	R/W	0BADh	T_AHCI_PORT_FBS_RX_PMP_REG_FIS_SPARE_31_16: NOTE: this field is reset by Cold Reset F0F0h: SPARE_31_16_INIT
15	RW1C	0h	T_AHCI_PORT_FBS_RX_PMP_REG_FIS_PMP15: 0h: PMP15_INIT (default) 1h: PMP15_CLR
14	RW1C	0h	T_AHCI_PORT_FBS_RX_PMP_REG_FIS_PMP14: 0h: PMP14_INIT (default) 1h: PMP14_CLR
13	RW1C	0h	T_AHCI_PORT_FBS_RX_PMP_REG_FIS_PMP13: 0h: PMP13_INIT (default) 1h: PMP13_CLR
12	RW1C	0h	T_AHCI_PORT_FBS_RX_PMP_REG_FIS_PMP12: 0h: PMP12_INIT (default) 1h: PMP12_CLR
11	RW1C	0h	T_AHCI_PORT_FBS_RX_PMP_REG_FIS_PMP11: 0h: PMP11_INIT (default) 1h: PMP11_CLR
10	RW1C	0h	T_AHCI_PORT_FBS_RX_PMP_REG_FIS_PMP10: 0h: PMP10_INIT (default) 1h: PMP10_CLR
9	RW1C	0h	T_AHCI_PORT_FBS_RX_PMP_REG_FIS_PMP9: 0h: PMP9_INIT (default) 1h: PMP9_CLR
8	RW1C	0h	T_AHCI_PORT_FBS_RX_PMP_REG_FIS_PMP8: 0h: PMP8_INIT (default) 1h: PMP8_CLR
7	RW1C	0h	T_AHCI_PORT_FBS_RX_PMP_REG_FIS_PMP7: 0h: PMP7_INIT (default) 1h: PMP7_CLR
6	RW1C	0h	T_AHCI_PORT_FBS_RX_PMP_REG_FIS_PMP6: 0h: PMP6_INIT (default) 1h: PMP6_CLR
5	RW1C	0h	T_AHCI_PORT_FBS_RX_PMP_REG_FIS_PMP5: 0h: PMP5_INIT (default) 1h: PMP5_CLR
4	RW1C	0h	T_AHCI_PORT_FBS_RX_PMP_REG_FIS_PMP4: 0h: PMP4_INIT (default) 1h: PMP4_CLR
3	RW1C	0h	T_AHCI_PORT_FBS_RX_PMP_REG_FIS_PMP3: 0h: PMP3_INIT (default) 1h: PMP3_CLR
2	RW1C	0h	T_AHCI_PORT_FBS_RX_PMP_REG_FIS_PMP2: 0h: PMP2_INIT (default) 1h: PMP2_CLR
1	RW1C	0h	T_AHCI_PORT_FBS_RX_PMP_REG_FIS_PMP1: 0h: PMP1_INIT (default) 1h: PMP1_CLR
0	RW1C	0h	T_AHCI_PORT_FBS_RX_PMP_REG_FIS_PMP0: 0h: PMP0_INIT (default) 1h: PMP0_CLR

T_AHCI_PORT_MP

Motion Protection

Offset: 0x17c | Read/Write: R/W | Reset: 0x00000002

Bit	R/W	Reset	Description
31:11	R/W	0	T_AHCI_PORT_MP_MPRSVD_2: 0h: MPRSVD_2_DEFAULT (default)
10	R/W	0	T_AHCI_PORT_MP_USE_EOF_DETECTION: 1h: USE_EOF_DETECTION_SET 0h: USE_EOF_DETECTION_RESET 0h: USE_EOF_DETECTION_DEFAULT (default)
9	R/W	0	T_AHCI_PORT_MP_COMRESET_ON_TIMEOUT: 1h: COMRESET_ON_TIMEOUT_SET 0h: COMRESET_ON_TIMEOUT_RESET 0h: COMRESET_ON_TIMEOUT_DEFAULT (default)
8	R/W	0	T_AHCI_PORT_MP_UNLOAD_WITH_COMRESET: 1h: UNLOAD_WITH_COMRESET_SET 0h: UNLOAD_WITH_COMRESET_RESET 0h: UNLOAD_WITH_COMRESET_DEFAULT (default)
7	R/W	0	T_AHCI_PORT_MP_MPA: 1h: MPA_SET 0h: MPA_CLEAR 0h: MPA_DEFAULT (default)
6	R/W	0	T_AHCI_PORT_MP_MPRSVD_1: 0h: MPRSVD_1_0 (default)
5	RW1C	0	T_AHCI_PORT_MP_MPIS: 0h: MPIS_ZERO (default) 1h: MPIS_SET 1h: MPIS_CLEAR
4	R	0	T_AHCI_PORT_MP_MPES: 1h: MPES_SET 0h: MPES_CLEAR 0h: MPES_DEFAULT (default)
3	R/W	0	T_AHCI_PORT_MP_MPRSVD_0: 0h: MPRSVD_0_0 (default)
2	R/W	0	T_AHCI_PORT_MP_MPSIE: 1h: MPSIE_SET 0h: MPSIE_CLEAR 0h: MPSIE_DEFAULT (default)
1	R/W	1	T_AHCI_PORT_MP_MPGBE: 1h: MPGBE_SET 0h: MPGBE_CLEAR 1h: MPGBE_DEFAULT (default)
0	R/W	0	T_AHCI_PORT_MP_MPE: 1h: MPE_SET 0h: MPE_CLEAR 0h: MPE_DEFAULT (default)

31.7.5 SATA and SATA0 FPCI Register Descriptions

This subsection describes the SATA and SATA0 FPCI registers. The SATA and SATA0 register sets are very similar. The register names and fields are shown as SATA<0> when both register sets are applicable.

For example, T_SATA<0>_CFG_0 defines two registers: T_SATA_CFG_0 and TSATA0_CFG_0. The register named T_SATA0_CFG_PHY_POWER is present only in the SATA0 register set.

See the Address Map section in this TRM for the mapping of both register sets.

31.7.5.1 PCI Configuration Registers

T_SATA<0>_CFG_0

PCI Vendor and Device ID Register

Offset: 0x00 | Read/Write: R | Reset: 0x0F70XXXX

Bit	R/W	Reset	Description
31:16	R	0F70h	T_SATA<0>_CFG_0_DEVICE_ID_UNIT: The DEVICE_ID bits identify the particular device. This identifier is allocated by the vendor. DEVICE_ID_FUNC bits contain the function number from the Configuration Address bits 10-8. DEVICE_ID_UNIT bits contain the Unit number within the chip. The Unit and Function are defined by parameters per block. D84h: DEVICE_ID_UNIT_MCP89 D84h: DEVICE_ID_UNIT_MCP89_PATA_DT D85h: DEVICE_ID_UNIT_MCP89_PATA_NB D88h: DEVICE_ID_UNIT_MCP89_AHCI_DT D89h: DEVICE_ID_UNIT_MCP89_AHCI_NB 580h: DEVICE_ID_UNIT_MCP89_AHCI_LINUX D8Ch: DEVICE_ID_UNIT_MCP89_RAID_DT D8Dh: DEVICE_ID_UNIT_MCP89_RAID_NB F70h: DEVICE_ID_UNIT_SATA (default)
15:0	R	10DEh	T_SATA<0>_CFG_0_VENDOR_ID: The VENDOR_ID bits identify the manufacturer of the device. Valid vendor identifiers are allocated by the PCI SIG to ensure uniqueness. VENDOR_ID is the joint SGS/NVIDIA PCI vendor ID. 10DEh: VENDOR_ID_NVIDIA

T_SATA<0>_CFG_1

PCI Device Control Register

The Device Control Command register provides coarse control over a device's ability to generate and respond to PCI cycles.

Offset: 0x04 | Read/Write: R/W | Reset: 0xXXXX0XXX

Bit	R/W	Reset	Description
31	R	0h	T_SATA<0>_CFG_1_DETECTED_PERR: The DETECTED_PERR bit indicates that the device has detected a parity error, even if parity error handing is disabled. (Bit 6 - T_SATA<0>_FPCI_PERR_DISABLED) 0h: DETECTED_PERR_NOT_ACTIVE
30	RW1C	0	T_SATA<0>_CFG_1_SIGNALED_SERR: The SIGNALED_SERR bit indicates that the device has asserted SERR#. NOTE: this field is reset by Cold Reset 0h: SIGNALED_SERR_NOT_ACTIVE (default) 1h: SIGNALED_SERR_ACTIVE 1h: SIGNALED_SERR_CLEAR
29	RW1C	0	T_SATA<0>_CFG_1_RECEIVED_MASTER: The RECEIVED_MASTER bit indicates that a master device's transaction (except for Special Cycle) was terminated with a master-abort. This means that no device on the PCI bus responded to the address of the mastered transaction. All master devices must implement this bit. When this bit is set, an interrupt is signaled in the PBUS_INTR_0 register. NOTE: this field is reset by Cold Reset 0h: RECEIVED_MASTER_NO_ABORT (default) 1h: RECEIVED_MASTER_ABORT 1h: RECEIVED_MASTER_CLEAR
28	RW1C	0	T_SATA<0>_CFG_1_RECEIVED_TARGET: The RECEIVED_TARGET bit indicates that a master device's transaction was terminated with a target-abort. All master devices must implement this bit. When this bit is set, an interrupt is signaled in the PBUS_INTR_0 register.

Bit	R/W	Reset	Description
			NOTE: this field is reset by Cold Reset 0h: RECEIVED_TARGET_NO_ABORT (default) 1h: RECEIVED_TARGET_ABORT 1h: RECEIVED_TARGET_CLEAR
27	R	0h	T_SATA<0>_CFG_1_SIGNALED_TARGET: The SIGNALED_TARGET bit indicates that the device has terminated a transaction with target-abort. Devices that will never signal target-abort do not need to implement this bit. When this bit is set, an interrupt is signaled in the PBUS_INTR_0 register. 0h: SIGNALED_TARGET_NO_ABORT
26:25	R	0h	T_SATA<0>_CFG_1_DEVSEL_TIMING: The DEVSEL_TIMING bits contain the timing of DEVSEL#. There are three allowable timings for assertion of DEVSEL#. These are encoded as 00b for fast, 01b for medium, and 10b for slow (11b is reserved). These bits are read only and must indicate the slowest time that a device asserts DEVSEL# for any bus command except Configuration Read and Configuration Write. Positive decode devices are required to respond with fast DEVSEL# (0-cycle). Only the subtractive function will respond with medium DEVSEL# to accept the cycle for the subtractive bus. 0h: DEVSEL_TIMING_FAST 1h: DEVSEL_TIMING_MEDIUM 2h: DEVSEL_TIMING_SLOW
24	R	0h	T_SATA<0>_CFG_1_MASTER_DATA_PERR: 0h: MASTER_DATA_PERR_NOT_ACTIVE
23	R	1h	T_SATA<0>_CFG_1_FAST_BACK2BACK: The FAST_BACK2BACK bit indicates that the device is capable of handling back-to-back transfers when the transactions are not to the same agent. This bit can be set to 1 if the device can accept these transactions, and must be set to 0 otherwise. 1h: FAST_BACK2BACK_CAPABLE 0h: FAST_BACK2BACK_INCAPABLE
22	R	0	Reserved
21	R	1h	T_SATA<0>_CFG_1_66MHZ: The 66MHZ bit indicates that the device is capable of 66 MHz PCI Bus operation. This value is initialized by a strapping bit. 1h: 66MHZ_CAPABLE 0h: 66MHZ_INCAPABLE
20	R	1h	T_SATA<0>_CFG_1_CAPLIST: The CAPLIST bit indicates that the device configuration space includes a capabilities list starting at the offset indicated by T_SATA<0>_CFG_13. 1h: CAPLIST_PRESENT 0h: CAPLIST_NOT_PRESENT
19	R	None	T_SATA<0>_CFG_1_INTR_STATUS: The INTR_STATUS bit is read-only and reflects the state of the interrupt in the device/function. Only when the INTR_DISABLE bit in the command register is a 0 and this INTR_STATUS bit is a 1, will the device's/function's INTx# singal be asserted. Setting the INTR_DISABLE bit to 1 has no effect on the state of this bit. 0h: INTR_STATUS_0 1h: INTR_STATUS_1
18:11	R	0	Reserved
10	R/W	0	T_SATA<0>_CFG_1_INTR_DISABLE: The INTR_DISABLE bit indicates that it could disable the device/function from asserting INTx#. A value of 0 enables the assertion of INTx#, and a value of 1 disables the assertion of INTx# signal. The Device Status register is used to record status information for PCI bus related events. 0h: INTR_DISABLE_ON (default) 1h: INTR_DISABLE_OFF
9	R	0h	T_SATA<0>_CFG_1_BACK2BACK: 0h: BACK2BACK_DISABLED 1h: BACK2BACK_ENABLED
8	R/W	0	T_SATA<0>_CFG_1_SERR: 0h: SERR_DISABLED (default)

Bit	R/W	Reset	Description
			1h: SERR_ENABLED
7	R	0h	T_SATA<0>_CFG_1_STEP: 0h: STEP_DISABLED 1h: STEP_ENABLED
6	R	0h	T_SATA<0>_CFG_1_PERR: 0h: PERR_DISABLED 1h: PERR_ENABLED
5	R	0h	T_SATA<0>_CFG_1_PALETTE_SNOOP: The PALETTE_SNOOP bit indicates that VGA compatible devices should snoop their palette registers. When this bit is set, special palette snooping behavior is enabled (ie, device must not respond). When the bit is reset, the device should treat palette accesses like all other accesses. VGA compatible devices should implement this bit. PALETTE_SNOOP is writable. 0h: PALETTE_SNOOP_DISABLED 1h: PALETTE_SNOOP_ENABLED
4	R	0h	T_SATA<0>_CFG_1_WRITE_AND_INVAL: The WRITE_AND_INVAL bit indicates that the device can use the Memory Write and Invalidate command when the transfer is aligned and 16 bytes and the contents of the Cache Line Size Register is 4 DWORDS. When this bit is 1, masters may generate the command. When it is 0, Memory Write must be used instead. State after RST# is 0. This bit must be implemented by master devices that can generate the Memory Write and Invalidate command. 0h: WRITE_AND_INVAL_DISABLED 1h: WRITE_AND_INVAL_ENABLED
3	R	0h	T_SATA<0>_CFG_1_SPECIAL_CYCLE: 0h: SPECIAL_CYCLE_DISABLED 1h: SPECIAL_CYCLE_ENABLED
2	R/W	0	T_SATA<0>_CFG_1_BUS_MASTER: The BUS_MASTER bit indicates that the device can act as a master on the PCI bus. A value of 0 disables the device from generating PCI accesses. A value of 1 allows the device to behave as a bus master. BUS_MASTER is writable. 0h: BUS_MASTER_DISABLED (default) 1h: BUS_MASTER_ENABLED
1	R/W	0	T_SATA<0>_CFG_1_MEMORY_SPACE: The MEMORY_SPACE bit indicates that the device will respond to memory space accesses. A value of 0 disables the device response. A value of 1 allows the device to respond to Memory space accesses. MEMORY_SPACE is writable. 0h: MEMORY_SPACE_DISABLED (default) 1h: MEMORY_SPACE_ENABLED
0	R/W	0	T_SATA<0>_CFG_1_IO_SPACE: The IO_SPACE bit indicates that the device will respond to I/O space accesses. A value of 0 disables the device response. A value of 1 allows the device to respond to I/O space accesses. IO_SPACE is writable. 0h: IO_SPACE_DISABLED (default) 1h: IO_SPACE_ENABLED

T_SATA<0>_CFG_2

PCI Revision ID and Class Code Register

This register provides a way to configure the SATA channel to operate in either compatibility or native PCI mode.

Offset: 0x08 | Read/Write: R | Reset: 0xXXXXxxA1

Bit	R/W	Reset	Description
31:16	R	101h	T_SATA<0>_CFG_2_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 (at offset 0BH) is a base class code which broadly classifies the type of function the device performs. The middle-byte (at offset 0AH)is a sub-class code which identifies more specificly the function of the device. The lower byte (at offset 09H) identifies a

Bit	R/W	Reset	Description
			specific register-level programming interface (if any) so that device independent software can interact with the device. 101h: CLASS_CODE_0101
15	R	1h	T_SATA<0>_CFG_2_SATA_BUS_MASTER: This bit defines the Bus mastering capability of SATA controller. SATA implements a back door register which can be used to change the class code. This register is T_SATA<0>_CFG_38_BKDOOR_CLASS_CODE. This provides BIOS a way to change the class code. 1h: SATA_BUS_MASTER_YES
14:12	R	0	Reserved
11	R	None (SATA) 0h (SATA0)	T_SATA<0>_CFG_2_SEC_PROG_IND: Secondary Programmable Indicator. Set to indicate it is programmable and can operate in both compatibility or native PCI mode. 0h: SEC_PROG_IND_NO (SATA0 only)
10	R	1 (SATA0)	T_SATA0_CFG_2_SEC_OP_MODE: Secondary Operating Mode 0 = Compatibility Mode (Use Legacy Addresses) 1 = Native Mode (Use Addresses Define in PCI I/O BARs) 1h: SEC_OP_MODE_INIT (default) (SATA0 only) 0h: SEC_OP_MODE_COMP 1h: SEC_OP_MODE_NTV
9	R	None (SATA) 0h (SATA0)	T_SATA<0>_CFG_2_PRI_PROG_IND: Primary Programmable Indicator. Set to indicate it is programmable and can operate in both compatibility or native PCI mode. 0h: PRI_PROG_IND_NO
8	R	1 (SATA0)	T_SATA0_CFG_2_PRI_OP_MODE: Primary Operating Mode 0 = Compatibility Mode (Use Legacy Addresses) 1 = Native Mode (Use Addresses Define in PCI I/O BARs) 1h: PRI_OP_MODE_INIT (default): (SATA0 only) 0h: PRI_OP_MODE_COMP 1h: PRI_OP_MODE_NTV
7:0	R	A1h	T_SATA<0>_CFG_2_REVISION_ID: The REVISION_ID bits specify a device specific revision identifier. The value is chosen by the vendor. Zero is an acceptable value. This field should be viewed as a vendor defined extension to the DEVICE_ID. A1h: REVISION_ID_VAL (default) A1h: REVISION_ID_A1 A2h: REVISION_ID_A2

T_SATA<0>_CFG_3

PCI Configuration Register

Offset: 0x0c | Read/Write: R/W | Reset: 0x00XXXXXX

Bit	R/W	Reset	Description
31:24	R	0	Reserved
23	R	0h	T_SATA<0>_CFG_3_HEADER_TYPE_FUNC: 0h: HEADER_TYPE_FUNC_SINGLE 1h: HEADER_TYPE_FUNC_MULTI
22:16	R	0h	T_SATA<0>_CFG_3_HEADER_TYPE_DEVICE: The HEADER_TYPE bits identify the layout of the bytes 10h: through 3FH in configuration space and also whether or not the device contains multiple functions. Bit 7 in this register is used to identify a multi-function device. If the bit is 0, then the device is single function. If the bit is 1, then the device has multiple functions. Bits 6 through 0 specify the layout of bytes 10h: through 3Fh. The LATENCY_TIMER and HEADER_TYPE are defined by parameters per block. 0h: HEADER_TYPE_DEVICE_NON_BRIDGE 1h: HEADER_TYPE_DEVICE_P2P_BRIDGE

Bit	R/W	Reset	Description
15:11	R	0h	T_SATA<0>_CFG_3_LATENCY_TIMER: The LATENCY_TIMER bits contain, in units of PCI bus clocks, the value of the Latency Timer for this PCI bus master. This register must be implemented as writable by any master that can burst more than two data phases. This register may be implemented as read-only for devices that burst two or fewer data phases, but the hardwired value must be limited to 16 or less. A typical implementation would be to build the five high-order bits (leaving the bottom three as read-only), resulting in a timer granularity of eight clocks. At reset, the register should be set to 0 (if programmable). LATENCY_TIMER bits are writable. 0h: LATENCY_TIMER_0_CLOCKS 1h: LATENCY_TIMER_8_CLOCKS 1Eh: LATENCY_TIMER_240_CLOCKS 1Fh: LATENCY_TIMER_248_CLOCKS
10:8	R	0	Reserved
7:0	R	0h	T_SATA<0>_CFG_3_CACHE_LINE_SIZE: 0h: CACHE_LINE_SIZE_0 20h: CACHE_LINE_SIZE_32 40h: CACHE_LINE_SIZE_64

T_SATA<0>_CFG_4

PCI Configuration Register

Offset: 0x10 | Read/Write: R/W | Reset: 0x0000000X (SATA) / 0x00000001 (SATA0)

Bit	R/W	Reset	Description
31:3	R/W	0	T_SATA<0>_CFG_4_BASE_ADDRESS: The BASE_ADDRESS bits contain the base address of the device. The number of upper bits that a device actually implements depends on how much of the address space the device will respond to. A device that wants a 1 MB memory address space (using a 32-bit base address register) would build the top 12 bits of the address register, hardwiring the other bits to 0. Power-up software can determine how much address space the device requires by writing a value of all 1's to the register and then reading the value back. The device will return 0's in all address bits which are not required, effectively specifying the address space required. When T_SATA<0>_CFG_2_PRI_OP_MODE is cleared, T_SATA_CFG_4 and T_SATA_CFG_5 will always read 32'h00000000. And when T_SATA<0>_CFG_2_SEC_OP_MODE is cleared, T_SATA<0>_CFG_6 and T_SATA_CFG_7 will always read 32'h00000000. 0h: BASE_ADDRESS_00 (default) 0h: BASE_ADDRESS_NTV_PCA
2:1	R	0	Reserved
0	R	n/a (SATA) 1 (SATA0)	T_SATA<0>_CFG_4_SPACE_TYPE: The SPACE_TYPE bit indicates whether the register maps into Memory or I/O space. 0h: SPACE_TYPE_MEMORY (SATA0 only) 1h: SPACE_TYPE_IO (default) (SATA0 only)

T_SATA<0>_CFG_5

PCI Configuration Register

Offset: 0x14 | Read/Write: R/W | Reset: 0x0000000X (SATA) / 0x00000001 (SATA0)

Bit	R/W	Reset	Description
31:2	R/W	0	T_SATA<0>_CFG_5_BASE_ADDRESS: The BASE_ADDRESS bits contain the base address of the device. The number of upper bits that a device actually implements depends on how much of the address space the device will respond to. A device that wants a 1 MB memory address space (using a 32-bit base address register) would build the top 12 bits of the address register, hardwiring the other bits to 0. Power-up software can determine how much address space the device requires by writing a value of all 1's to the register and then reading the value back. The device will return 0's in all address bits which are not required, effectively specifying the address space required. When T_SATA<0>_CFG_2_PRI_OP_MODE is cleared, T_SATA_CFG_4 and T_SATA_CFG_5 will always read 32'h00000000. And when T_SATA<0>_CFG_2_SEC_OP_MODE is cleared,

Bit	R/W	Reset	Description
			T_SATA<0>_CFG_6 and T_SATA<0>_CFG_7 will always read 32'h00000000. 0h: BASE_ADDRESS_00 (default) 0h: BASE_ADDRESS_NTV_PSA
1	R	0	Reserved
0	R	n/a (SATA) 1 (SATA0)	T_SATA<0>_CFG_5_SPACE_TYPE: The SPACE_TYPE bit indicates whether the register maps into Memory or I/O space. 0h: SPACE_TYPE_MEMORY (SATA0 only) 1h: SPACE_TYPE_IO (default) (SATA0 only)

T_SATA<0>_CFG_6

PCI Configuration Register

Offset: 0x18 | Read/Write: R/W | Reset: 0x0000000X (SATA) / 0x00000001 (SATA0)

Bit	R/W	Reset	Description
31:3	R/W	0	T_SATA<0>_CFG_6_BASE_ADDRESS: The BASE_ADDRESS bits contain the base address of the device. The number of upper bits that a device actually implements depends on how much of the address space the device will respond to. A device that wants a 1 MB memory address space (using a 32-bit base address register) would build the top 12 bits of the address register, hardwiring the other bits to 0. Power-up software can determine how much address space the device requires by writing a value of all 1's to the register and then reading the value back. The device will return 0's in all address bits which are not required, effectively specifying the address space required. When T_SATA<0>_CFG_2_PRI_OP_MODE is cleared, T_SATA_CFG_4 and T_SATA_CFG_5 will always read 32'h00000000. And when T_SATA<0>_CFG_2_SEC_OP_MODE is cleared, T_SATA<0>_CFG_6 and T_SATA_CFG_7 will always read 32'h00000000. 0h: BASE_ADDRESS_00 (default) 0h: BASE_ADDRESS_NTV_SCA
2:1	R	0	Reserved
0	R	n/a (SATA) 1 (SATA0)	T_SATA<0>_CFG_6_SPACE_TYPE: The SPACE_TYPE bit indicates whether the register maps into Memory or I/O space. 0h: SPACE_TYPE_MEMORY (SATA0 only) 1h: SPACE_TYPE_IO (default) (SATA0 only)

T_SATA<0>_CFG_7

PCI Configuration Register

Offset: 0x1c | Read/Write: R/W | Reset: 0x0000000X (SATA) / 0x00000001 (SATA0)

Bit	R/W	Reset	Description
31:2	R/W	0	T_SATA<0>_CFG_7_BASE_ADDRESS: The BASE_ADDRESS bits contain the base address of the device. The number of upper bits that a device actually implements depends on how much of the address space the device will respond to. A device that wants a 1 MB memory address space (using a 32-bit base address register) would build the top 12 bits of the address register, hardwiring the other bits to 0. Power-up software can determine how much address space the device requires by writing a value of all 1's to the register and then reading the value back. The device will return 0's in all address bits which are not required, effectively specifying the address space required. When T_SATA<0>_CFG_2_SEC_OP_MODE is cleared, this register will always read 32'h00000000. 0h: BASE_ADDRESS_00 (default) 0h: BASE_ADDRESS_NTV_SSA
1	R	0	Reserved
0	R	n/a (SATA) 1 (SATA0)	T_SATA<0>_CFG_7_SPACE_TYPE: The SPACE_TYPE bit indicates whether the register maps into Memory or I/O space. 0h: SPACE_TYPE_MEMORY (SATA0 only) 1h: SPACE_TYPE_IO (default) (SATA0 only)

T_SATA<0>_CFG_8

PCI Configuration Register

T_SATA<0>_CFG_8 is a 16-byte I/O BAR for two things: - The Bus Master control registers for both channels.

Offset: 0x20 | Read/Write: R/W | Reset: 0x0000000X

Bit	R/W	Reset	Description
31:4	R/W	0	T_SATA<0>_CFG_8_BASE_ADDRESS: The BASE_ADDRESS bits contain the base address of the device. The number of upper bits that a device actually implements depends on how much of the address space the device will respond to. A device that wants a 1 MB memory address space (using a 32-bit base address register) would build the top 12 bits of the address register, hardwiring the other bits to 0. Power-up software can determine how much address space the device requires by writing a value of all 1's to the register and then reading the value back. The device will return 0's in all address bits which are not required, effectively specifying the address space required. 0h: BASE_ADDRESS_BMA (default)
3:1	R	0	Reserved
0	R	1h	T_SATA<0>_CFG_8_SPACE_TYPE: The SPACE_TYPE bit indicates whether the register maps into Memory or I/O space. 1h: SPACE_TYPE_IO 0h: SPACE_TYPE_MEMORY

T_SATA<0>_CFG_9

PCI Memory BAR for AHCI Register

T_SATA<0>_CFG_9 is memory-BAR register for AHCI registers. Size of this memory space is 4KB+256Bytes. First 32 Bytes are used for Host control registers. Next 128 bytes are reserved. Next 96 bytes for vendor specific registers. Each port has 128-byte register space.

Offset: 0x24 | Read/Write: R/W | Reset: 0x0000000X

Bit	R/W	Reset	Description
31:13	R/W	0	T_SATA<0>_CFG_9_BASE_ADDRESS: 0h: BASE_ADDRESS_INIT (default)
12:1	R	0	Reserved
0	R	0h	T_SATA<0>_CFG_9_SPACE_TYPE: 1h: SPACE_TYPE_IO 0h: SPACE_TYPE_MEMORY

T_SATA<0>_CFG_10

PCI Configuration Register

These are unused BAR locations.

Offset: 0x28 | Read/Write: R | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:0	R	0h	T_SATA<0>_CFG_10_RESERVED: 0h: RESERVED_0

T_SATA<0>_CFG_11

PCI Subsystem Vendor ID and Subsystem ID Register

Offset: 0x2c | Read/Write: R | Reset: 0x00000000

Bit	R/W	Reset	Description
31:16	R	0	T_SATA<0>_CFG_11_SUBSYSTEM_ID: Subsystem ID is read-only. The system BIOS can set this value by writing to Configuration [42h]. 0h: SUBSYSTEM_ID_NONE (default)
15:0	R	0	T_SATA<0>_CFG_11_SUBSYSTEM_VENDOR_ID: Subsystem Vendor ID is read-only. The system BIOS can set this value by writing to Configuration [40h]. 0h: SUBSYSTEM_VENDOR_ID_NONE (default)

T_SATA<0>_CFG_12

Expansion ROM Base Address Configuration Register

The Expansion ROM Base Address configuration register should not be used for any PCI integrated blocks.

Offset: 0x30 | Read/Write: R | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:0	R	0h	T_SATA<0>_CFG_12_RESERVED: 0h: RESERVED_0

T_SATA<0>_CFG_13

PCI Capability Pointer Register

Offset: 0x34 | Read/Write: R/W | Reset: 0x000000XX

Bit	R/W	Reset	Description
31:8	R	0	Reserved
7:0	R	44h	T_SATA<0>_CFG_13_CAP_PTR: The CAP_PTR bits indicate the offset into configuration space where the capabilities list begins. This always points to 0x44 where at least the PCI-PM registers are expected to reside. 44h: CAP_PTR_PCIPM

T_SATA<0>_CFG_14

PCI Configuration Register

Offset: 0x38 | Read/Write: R | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:0	R	0h	T_SATA<0>_CFG_14_RESERVED: The RESERVED bits are reserved for future use. 0h: RESERVED_0

T_SATA<0>_CFG_15

PCI Configuration Register

Offset: 0x3c | Read/Write: R/W | Reset: 0xXXXXXX00 (SATA) / 0xXXXX0100 (SATA0)

Bit	R/W	Reset	Description
31:24	R	1h	T_SATA<0>_CFG_15_MAX_LAT: The MAX_LAT bits contain the maximum time the device requires to gain access to the PCI bus. This read-only register is used to specify the device's desired settings for Latency Timer values. The value specifies a period of time in units of 1/4 microsecond. Values of 0 indicate that the device has no major requirements for the settings of Latency Timers. MAX_LAT is nonzero.

Bit	R/W	Reset	Description
			The INTR_PIN, MIN_GNT, and MAX_LAT are configurable per block. 0h: MAX_LAT_NO_REQUIREMENTS 1h: MAX_LAT_250NS
23:16	R	3h	T_SATA<0>_CFG_15_MIN_GNT: The MIN_GNT bits contain the length of the burst period a device needs assuming a clock rate of 33 MHz. This read-only register is used to specify the device's desired settings for Latency Timer values. The value specifies a period of time in units of 1/4 microsecond. Values of 0 indicate that the device has no major requirements for the settings of Latency Timers. MIN_GNT is nonzero. 0h: MIN_GNT_NO_REQUIREMENTS 3h: MIN_GNT_750NS
15:8	R	1 (SATA0)	T_SATA0_CFG_15_INTR_PIN: The INTR_PIN bits contain the interrupt pin the device (or device function) uses. A value of 1 corresponds to INTA#. A value of 2 corresponds to INTB#. A value of 3 corresponds to INTC#. A value of 4 corresponds to INTD#. Devices (or device functions) that don't use an interrupt pin must put a 0 in this register. 0h: INTR_PIN_NONE (SATA0 only) 1h: INTR_PIN_INTA (default) (SATA0 only) 2h: INTR_PIN_INTB 3h: INTR_PIN_INTC 4h: INTR_PIN_INTD
7:0	R/W	0	T_SATA<0>_CFG_15_INTR_LINE: The INTR_LINE bits contain the interrupt routing information. The register is read/write and must be implemented by any device (or device function) that uses an interrupt pin. POST software will write the routing information into this register as it initializes and configures the system. The value in this register tells which input of the system interrupt controller(s) the device's interrupt pin is connected to. Device drivers and operating systems can use this information to determine priority and vector information. INTR_LINE is initialized to 0xff (no connection) at reset. Some PCI BIOS' can't handle aliased INTR_LINEs. If T_SATA<0>_CFG_2_PRI_OP_MODE or T_SATA<0>_CFG_2_SEC_OP_MODE are set, this register will read whatever value written by software. If both the bits are cleared the register will always return 00h: 0h: INTR_LINE_IRQ0 (default) 1h: INTR_LINE_IRQ1 Fh: INTR_LINE_IRQ15 FFh: INTR_LINE_UNKNOWN

31.7.5.2 Device Specific Configuration Registers

The CFG_16 through CFG_63 registers are Device Specific PCI configuration registers.

T_SATA<0>_CFG_16

Write Subsystem Vendor ID and Subsystem ID Register

Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:16	R/W	0	T_SATA<0>_CFG_16_SUBSYSTEM_ID: Subsystem ID write register. The system BIOS can set the Subsystem ID value in [2Eh] by writing to this register. 0h: SUBSYSTEM_ID_NONE (default)
15:0	R/W	0	T_SATA<0>_CFG_16_SUBSYSTEM_VENDOR_ID: Subsystem Vendor ID write register. The system BIOS can set the Subsystem Vendor ID value in [2Ch] by writing to this register. 0h: SUBSYSTEM_VENDOR_ID_NONE (default)

T_SATA<0>_CFG_17

PCI Power Management Capabilities Register

The CFG_17 through CFG_18 registers are for PCI Power Management operation. The ROM Based Power Data is not supported.

The CFG_17 register is the Capabilities Identification information register for the PCI PM functionality. NEXT_PTR is a byte offset which points to the next item in the capabilities list. CAP_ID indicates this capability's record type.

Offset: 0x44 | Read/Write: R/W | Reset: 0xXXXX8CXX

Bit	R/W	Reset	Description
31	R	0h	T_SATA<0>_CFG_17_PME_SUPPORT_D3C: 1h: PME_SUPPORT_D3C_YES 0h: PME_SUPPORT_D3C_NO
30	R	0h	T_SATA<0>_CFG_17_PME_SUPPORT_D3H: 1h: PME_SUPPORT_D3H_YES 0h: PME_SUPPORT_D3H_NO
29	R	0h	T_SATA<0>_CFG_17_PME_SUPPORT_D2: 1h: PME_SUPPORT_D2_YES 0h: PME_SUPPORT_D2_NO
28	R	0h	T_SATA<0>_CFG_17_PME_SUPPORT_D1: 1h: PME_SUPPORT_D1_YES 0h: PME_SUPPORT_D1_NO
27	R	0h	T_SATA<0>_CFG_17_PME_SUPPORT_D0: 1h: PME_SUPPORT_D0_YES 0h: PME_SUPPORT_D0_NO
26	R	0h	T_SATA<0>_CFG_17_D2_SUPPORT: 1h: D2_SUPPORT_YES 0h: D2_SUPPORT_NO
25	R	0h	T_SATA<0>_CFG_17_D1_SUPPORT: 1h: D1_SUPPORT_YES 0h: D1_SUPPORT_NO
24:22	R	0h	T_SATA<0>_CFG_17_AUX_CURRENT: 0h: AUX_CURRENT_0 1h: AUX_CURRENT_55MA 2h: AUX_CURRENT_100MA 3h: AUX_CURRENT_160MA 4h: AUX_CURRENT_220MA 5h: AUX_CURRENT_270MA 6h: AUX_CURRENT_320MA 7h: AUX_CURRENT_375MA
21	R	0h	T_SATA<0>_CFG_17_DEV_SPEC_INIT: 0h: DEV_SPEC_INIT_NOT_NEEDED 1h: DEV_SPEC_INIT_NEEDED
20	R	0	Reserved
19	R	0h	T_SATA<0>_CFG_17_PME_CLOCK: 0h: PME_CLOCK_NOT_NEEDED 1h: PME_CLOCK_NEEDED
18:16	R	2h	T_SATA<0>_CFG_17_PCIPM_REV: 2h: PCIPM_REV_11
15:8	R	8Ch	T_SATA<0>_CFG_17_NEXT_PTR: NOTE: this field is reset by Cold Reset B0h: NEXT_PTR_MSI C4h: NEXT_PTR_MSIX 8Ch: NEXT_PTR_SATA (default) 0h: NEXT_PTR_NULL
7:0	R	1h	T_SATA<0>_CFG_17_CAP_ID: 1h: CAP_ID_PM

T_SATA<0>_CFG_18

PCI Power Management Control/Status Register

Offset: 0x48 | Read/Write: R/W | Reset: 0x0000XX00

Bit	R/W	Reset	Description
31:16	R	0	Reserved
15	R	0h	T_SATA<0>_CFG_18_PME_STATUS: PME Status : The SATA function does not support PME generation. 0h: PME_STATUS_NOT_ACTIVE
14:9	R	0	Reserved
8	R	0h	T_SATA<0>_CFG_18_PME: PME Enable : The SATA function does not support PME generation. 0h: PME_DISABLE
7:2	R	0	Reserved
1:0	R/W	0	T_SATA<0>_CFG_18_PM_STATE: Power State : This is used to determine the current power state. Software updates this register when changing power states. 00b = D0, 01b = D1, 10b = D2, 11b = D3hot. 0h: PM_STATE_D0 (default) 3h: PM_STATE_D3

T_SATA<0>_FPCI_SW

Offset: 0x54 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:9	R	0h	T_SATA<0>_FPCI_SW_RSVD_9_31: 0h: RSVD_9_31_VAL
8	R/W	0	T_SATA<0>_FPCI_SW_WAKEUP_PLL: When the dev_clk PLL is shutdown, this field is written to one by SW to wakeup the PLL before issuing any other write 0h: WAKEUP_PLL_INIT (default)
7:1	R	0h	T_SATA<0>_FPCI_SW_RSVD_1_7: 0h: RSVD_1_7_VAL
0	R/W	0	T_SATA<0>_FPCI_SW_IDDQ_PG: SW writes this bit to one/ zero when Power Gated which drives the IDDQ to the pads Single bit is used to drive all port's IDDQ 0h: IDDQ_PG_INIT (default)

T_SATA<0>_ATACAP0

Serial ATA Capability Register 0

This is Serial ATA Capability register 0 as defined in the AHCI rev 1.1 specification.

Offset: 0x8c | Read/Write: R | Reset: 0xXXXXB0XX

Bit	R/W	Reset	Description
31:24	R	0h	T_SATA<0>_ATACAP0_RESERVED: 0h: RESERVED_0
23:20	R	1h	T_SATA<0>_ATACAP0_MAJREV: 1h: MAJREV_INIT
19:16	R	0h	T_SATA<0>_ATACAP0_MINREV: 0h: MINREV_INIT
15:8	R	B0h	T_SATA<0>_ATACAP0_NEXT:

Bit	R/W	Reset	Description
			NOTE: this field is reset by Cold Reset B0h: NEXT_MSI (default) C4h: NEXT_MSIX 0h: NEXT_NULL
7:0	R	12h	T_SATA<0>_ATACAP0_CID: 12h: CID_SATA

T_SATA<0>_ATACAP1

Serial ATA Capability Register 1

This is Serial ATA Capability register 1 as defined in the AHCI rev 1.1 specification.

Offset: 0x90 | Read/Write: R | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:16	R	0h	T_SATA<0>_ATACAP1_RESERVED: 0h: RESERVED_0
15:4	R	25h	T_SATA<0>_ATACAP1_BAROFST: 25h: BAROFST_94H
3:0	R	Fh	T_SATA<0>_ATACAP1_BARLOC: Fh: BARLOC_CFG

T_SATA<0>_CFG_35

Serial ATA IDP Index

This is Serial ATA IDP Index register as defined in the AHCI rev 1.1 specification.

Offset: 0x94 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:13	R/W	0	T_SATA<0>_CFG_35_RESERVED: 0h: RESERVED_0 (default)
12:2	R/W	0	T_SATA<0>_CFG_35_IDP_INDEX: 0h: IDP_INDEX_0 (default)
1:0	R/W	0	T_SATA<0>_CFG_35_RESERVED1: 0h: RESERVED1_0 (default)

T_SATA0_AHCI_IDP1

Serial ATA IDP DATA

This is the Serial ATA IDP Data register as defined in the AHCI rev 1.1 specification.

This register applies to the SATA0 register space only.

Offset: 0x98 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:0	R/W	None	T_SATA0_AHCI_IDP1_DATA

T_SATA<0>_MSI_CTRL

MSI Message Control and Capability Register

Following are registers required to support MSI interrupt modes.

int0 is either non-ADMA primary channel interrupt or primary channel ADMA interrupt if the command completes without any error. int1 is similarly non-ADMA secondary channel interrupt or secondary channel ADMA interrupt if the command completes without any error. int2 is used to signal primary channel erroneous command completion. int3 is used to signal secondary channel erroneous command completion.

MSI_CTRL is the main control register for MSI support.

Offset: 0xb0 | Read/Write: R/W | Reset: 0x008600XX

Bit	R/W	Reset	Description
31:25	R	0	Reserved
24	R	0h	T_SATA<0>_MSI_CTRL_VECTOR_MASK_CAP: The VECTOR_MASK_CAP field indicates whether or not the controller supports MSI-per-vector masking. 0h: VECTOR_MASK_CAP_DIS 1h: VECTOR_MASK_CAP_EN 0h: VECTOR_MASK_CAP_DEFAULT (default)
23	R	1h	T_SATA<0>_MSI_CTRL_64_ADDR_CAP: The 64_ADDR_CAP field indicates whether or not the controller is capable of generating a 64-bit message address. A value of 1 means the controller is capable of generating a 64-bit message address. 0h: 64_ADDR_CAP_DIS 1h: 64_ADDR_CAP_EN 1h: 64_ADDR_CAP_DEFAULT (default)
22:20	R/W	0	T_SATA<0>_MSI_CTRL_MULT_MSG_ENABLE: System software writes to this field to indicate the number of allocated vectors (less than or equal to the number of vectors requested). The number of vectors is aligned as a power of two. When MSI is enabled, the controller will be allocated at least one vector. 0h: MULT_MSG_ENABLE_1 1h: MULT_MSG_ENABLE_2 2h: MULT_MSG_ENABLE_4 3h: MULT_MSG_ENABLE_8 4h: MULT_MSG_ENABLE_16 5h: MULT_MSG_ENABLE_32 0h: MULT_MSG_ENABLE_DEFAULT (default)
19:17	R	3h	T_SATA<0>_MSI_CTRL_MULT_MSG_CAP: System software reads this field to determine the number of requested vectors. The number of requested vectors must be aligned to a power of two. Values of 6 and 7 in this field are reserved. 0h: MULT_MSG_CAP_1 1h: MULT_MSG_CAP_2 2h: MULT_MSG_CAP_4 3h: MULT_MSG_CAP_8 4h: MULT_MSG_CAP_16 5h: MULT_MSG_CAP_32 3h: MULT_MSG_CAP_DEFAULT (default)
16	R/W	0	T_SATA<0>_MSI_CTRL_MSI_ENABLE: The MSI_ENABLE field enables the MSI capability. If MSI_ENABLE is written to a 1, the controller is permitted to use MSI to request service and is prohibited from using the legacy interrupt. System configuration software sets this bit to enable MSI. A device driver is prohibited from writing this bit to mask the controller's service request. If this bit is written to a 0, the controller is prohibited from using MSI to request service. 0h: MSI_ENABLE_OFF 1h: MSI_ENABLE_ON 0h: MSI_ENABLE_DEFAULT (default)
15:8	R	0h	T_SATA<0>_MSI_CTRL_NEXT_PTR: The NEXT_PTR field identifies the next item in the capabilities list. It is a read-only field. 0h: NEXT_PTR_NULL ECh NEXT_PTR_MMAP 0h: NEXT_PTR_DEFAULT (default)
7:0	R	5h	T_SATA<0>_MSI_CTRL_CAP_ID: The CAP_ID field identifies this capability block as the MSI capability block. This is read-only as 0x5.

Bit	R/W	Reset	Description
			5h: CAP_ID_MSI

T_SATA<0>_MSI_ADDR1

MSI Message Address Register

MSI_ADDR1 specifies the lower 32 bits to which an MSI memory write transaction should occur.

Offset: 0xb4 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:2	R/W	0	T_SATA<0>_MSI_ADDR1_MSG_ADDR: System-specified message address. When MSI is enabled, this field specifies the dword-aligned address for the MSI memory write transaction. 0h: MSG_ADDR_DEFAULT (default)
1:0	R	0	Reserved

T_SATA<0>_MSI_ADDR2

MSI Message Upper Address Register

MSI_ADDR2 specifies the upper 32 bits to which an MSI memory write transaction should occur.

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

Bit	R/W	Reset	Description
31:0	R/W	0	T_SATA<0>_MSI_ADDR2_MSG_ADDR: System-specified message address. When MSI is enabled, this field specifies the upper 32-bits of the address for the MSI memory write transaction. The contents of this register only apply when T_SATA<0>_MSI_CTRL_64_ADDR_CAP bit is set. 0h: MSG_ADDR_DEFAULT (default)

T_SATA<0>_MSI_DATA

MSI Message Data Register

The MSI_DATA register contains the system-specified message.

Offset: 0xbc | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:16	R	0	Reserved
15:0	R/W	0	T_SATA<0>_MSI_DATA_MSG_DATA: System-specified message. When MSI is enabled, the message data is driven onto the lower 16 bits of the MSI memory write. The MULT_MSG_ENABLE field in configuration register B0h specifies the number of low order message data bits that the SATA is permitted to modify to generate its system software allocated vectors. 0h: MSG_DATA_DEFAULT (default)

T_SATA<0>_MSI_QUEUE

MSI Message Queue Configuration Register

The MSI_QUEUE register is a private register. It specifies to which virtual channel queue (ISO or NON-ISO) that the MSI message will be sent to before being sent out on FPCI bus.

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:4	R	0	Reserved
3:0	R/W	0	T_SATA<0>_MSI_QUEUE_Bit: This field specifies to which VC queue a particular MSI message will be sent. bit 0 corresponds to MSI vector 0 bit 1 corresponds to MSI vector 1 bit 2 corresponds to MSI vector 2 bit 3 corresponds to MSI vector 3 0h: Bit_DEFAULT (default)

T_SATA<0>_MSI_MAP

MSI Mapping Capability Register

The MSI_MAP register is used to tell the OS that MSIs are supported in K8 mode.

This is the per-device HT MSI Capability Block.

Offset: 0xec | Read/Write: R/W | Reset: 0xA80X00XX

Bit	R/W	Reset	Description
31:27	R	15h	T_SATA<0>_MSI_MAP_CAP_TYPE: The CAP_TYPE field is read only at 0x15 to indicate that this is an MSI Mapping Capability block. 15h: CAP_TYPE_DEFAULT (default)
26:18	R	0	Reserved
17	R	1h	T_SATA<0>_MSI_MAP_FIXD: The FIXD bit is a read-only bit indicating if the next 2 dwords for programmable address are present in the capability. If set, the address for mapping MSIs is fixed at 0x0000_0000_FEEx_xxxx and that this capability block is 1 dword long. 1h: FIXD_ON
16	R/W	0	T_SATA<0>_MSI_MAP_EN: The EN bit indicates if the mapping is active. It is cleared upon warm reset. 0h: EN_OFF 1h: EN_ON 0h: EN_DEFAULT (default)
15:8	R	0h	T_SATA<0>_MSI_MAP_NEXT_PTR: The NEXT_PTR field points to the next item in the capabilities list. It is a read-only field. 0h: NEXT_PTR_NULL 0h: NEXT_PTR_DEFAULT (default)
7:0	R	8h	T_SATA<0>_MSI_MAP_CAP_ID: The CAP_ID field identifies that this is an HT capability list item. (Read-only as 0x8). 8h: CAP_ID_MSI

T_SATA<0>_INDIRECT_IDP0

IDP pair to access 257-4K address space - Address register

Offset: 0xf0 | Read/Write: R/W | Reset: 0xXXXXX00X

Bit	R/W	Reset	Description
31:12	R	0h	T_SATA<0>_INDIRECT_IDP0_RESERVED: 0h: RESERVED_0
11:2	R/W	0	T_SATA<0>_INDIRECT_IDP0_IDP_INDEX: 0h: IDP_INDEX_0 (default)
1:0	R	0h	T_SATA<0>_INDIRECT_IDP0_RESERVED1: 0h: RESERVED1_0

T_SATA<0>_INDIRECT_IDP1

IDP pair to access 257-4K address space - DATA register

Offset: 0xf4 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:0	R/W	None	T_SATA<0>_INDIRECT_IDP1_DATA:

T_SATA<0>_FPCICFG

FPCI Debug register

The FPCICFG register is for the configuration bits that the FPCI wrapper needs.

Offset: 0xf8 | Read/Write: R/W | Reset: 0x008CC000

Bit	R/W	Reset	Description
31:28	R/W	0	T_SATA<0>_FPCICFG_DEVID_OVERRIDE_ID: Use this 4 bit register to program a desired value between 0x580-0x58F on a linux system and between 0x550-0x55B on a no linux system. The default values for NBP and non-NBP systems are as listed below. Non-NBP systems cannot program values of 0x2, 0x4 and 0xA. If one tried to program a value outside the limits listed here, then the last value of this field is retained. 0h: DEVID_OVERRIDE_ID_DEFAULT (default) 0h: DEVID_OVERRIDE_ID_DEFAULT_NBP
27	R/W	0	T_SATA<0>_FPCICFG_DEVID_OVERRIDE_ENABLE: This is bit when set indicates a Linux system. 0h: DEVID_OVERRIDE_ENABLE_OFF (default) 0h: DEVID_OVERRIDE_ENABLE_ON
26	R/W	0	T_SATA<0>_FPCICFG_DROP_ON_TA_ERR_ENABLE: 0h: DROP_ON_TA_ERR_ENABLE_DEFAULT (default) 0h: DROP_ON_TA_ERR_ENABLE_OFF 1h: DROP_ON_TA_ERR_ENABLE_ON
25	R/W	0	T_SATA<0>_FPCICFG_DROP_ON_MA_ERR_ENABLE: 0h: DROP_ON_MA_ERR_ENABLE_DEFAULT (default) 0h: DROP_ON_MA_ERR_ENABLE_OFF 1h: DROP_ON_MA_ERR_ENABLE_ON
24	R/W	0	T_SATA<0>_FPCICFG_DROP_ON_ERR_ENABLE: 0h: DROP_ON_ERR_ENABLE_DEFAULT (default) 0h: DROP_ON_ERR_ENABLE_OFF 1h: DROP_ON_ERR_ENABLE_ON
23	R/W	1	T_SATA<0>_FPCICFG_FIX_DEADLOCK_ENABLE: 1h: FIX_DEADLOCK_ENABLE_DEFAULT (default) 0h: FIX_DEADLOCK_ENABLE_OFF 1h: FIX_DEADLOCK_ENABLE_ON
22	R/W	0	T_SATA<0>_FPCICFG_PASSIVE_UID_CLMP_ENABLE: 1h: PASSIVE_UID_CLMP_ENABLE_SET 0h: PASSIVE_UID_CLMP_ENABLE_CLR (default)
21	R	0	T_SATA<0>_FPCICFG_PASSIVE_UID_CLMP: 1h: PASSIVE_UID_CLMP_SUPP 0h: PASSIVE_UID_CLMP_NOT_SUPP 0h: PASSIVE_UID_CLMP_DEFAULT (default)
20:16	R/W	0Ch	T_SATA<0>_FPCICFG_NONISO_READ_CREDITS: Ch NONISO_READ_CREDITS_DEFAULT (default) 0h: NONISO_READ_CREDITS_MIN 1Fh NONISO_READ_CREDITS_MAX
15	R/W	1	T_SATA<0>_FPCICFG_ERR_SEVERITY: 1h: ERR_SEVERITY_DEFAULT (default) 0h: ERR_SEVERITY_NONFATAL

Bit	R/W	Reset	Description
			1h: ERR_SEVERITY_FATAL
14	R/W	1	T_SATA<0>_FPCICFG_TGTDONE_PASSPW: 1h: TGTDONE_PASSPW_DEFAULT (default) 1h: TGTDONE_PASSPW_SET 0h: TGTDONE_PASSPW_CLR
13:9	R	0	Reserved
8	R/W	0	T_SATA<0>_FPCICFG_CREDIT_SYS_ENABLE: FPCICFG_CREDIT_SYS_ENABLE is used to enable the read credit system. This is a performance feature. When this bit is set to OFF, the wrapper will assert fpci2dev_busy when the toy has reached its maximum number of outstanding NONISO or ISO read requests (determined by wrapper parameterization). When this bit is set to ON, the wrapper will only throttle toy mastered requests when there is contention for wrapper resources. This allows the toy bus master to issue writes even when the read credits have been exhausted. The T_SATA<0>_FPCICFG_ISO_READ_CREDITS field is used when the read credit system is enabled. This field sets the maximum number of outstanding ISO read requests allowed by the toy. The value in this field must NOT exceed ISO_READ_CREDITS_MAX. The T_SATA<0>_FPCICFG_NONISO_READ_CREDITS field is used when the read credit system is enabled. This field sets the maximum number of outstanding NONISO read requests allowed by the toy. The value in this field must NOT exceed NONISO_READ_CREDITS_MAX. 0h: CREDIT_SYS_ENABLE_DEFAULT (default) 0h: CREDIT_SYS_ENABLE_OFF 1h: CREDIT_SYS_ENABLE_ON
7:6	R/W	0	T_SATA<0>_FPCICFG_COHCMD: 0h: COHCMD_DEFAULT (default) 0h: COHCMD_TOY 1h: COHCMD_COH 2h: COHCMD_NONCOH
5:4	R/W	0	T_SATA<0>_FPCICFG_RSPPASSPW: FPCICFG_RSPPASSPW indicates that for a master command issued from the wrapper to the UFA, it's issued with RSPPASSPW set, RSPPASSPW not set, or whatever it is sent with from the toy. These bits only influence non-posted commands. 0h: RSPPASSPW_DEFAULT (default) 0h: RSPPASSPW_TOY 1h: RSPPASSPW_PASS 2h: RSPPASSPW_NOPASS
3:2	R/W	0	T_SATA<0>_FPCICFG_PASSPW: FPCICFG_PASSPW indicates that for a master command issued from the wrapper to the UFA, it's issued with PASSPW set, PASSPW not set, or whatever it is sent with from the toy. These bits only influence non-broadcast commands. 0h: PASSPW_DEFAULT (default) 0h: PASSPW_TOY 1h: PASSPW_PASS 2h: PASSPW_NOPASS
1:0	R/W	0	T_SATA<0>_FPCICFG_ISOCMD: T_IDE_FPCICFG_ISOCMD indicates that for a master command issued from the wrapper to the UFA, it's issued as ISO, NONISO, or whatever it is sent as from the toy. These bits only influence commands that have ISO/NONISO counterparts. 0h: ISOCMD_DEFAULT (default) 0h: ISOCMD_TOY 1h: ISOCMD_ISO 2h: ISOCMD_NONISO

T_SATA<0>_SCRATCH_1

PCI Configuration Register

This general purpose scratch register is used for communication between the storage SW and the SBIOS.

This field is designed to be used to allow the SBIOS to communicate to the storage SW which ports are enabled for RAID operation.

Offset: 0xfc | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	T_SATA<0>_SCRATCH_1_SBIOS: 0h: SBIOS_DEFAULT (default)

T_SATA<0>_NVOOB

Serial ATA internal PHY control register used in nvoob

20 + 2*ACTIVE_CNT_HIGH is the max number of 300 MHz cycles squelch can be high for OOB signaling. 20 + 2*ACTIVE_CNT_LOW is the minimum number of 300 MHz cycles squelch can be high for OOB signaling. 100 + 4*COMINIT_IDLE_CNT_HIGH is the max number of 300 MHz cycles squelch can be low for cominit signaling. 50 + 4*COMINIT_IDLE_CNT_LOW is the min number of 300 MHz cycles squelch can be low for cominit signaling. 32 + 2*COMWAKE_IDLE_CNT_HIGH is the max number of 300 MHz cycles squelch can be low for comwake signaling. 2*COMWAKE_IDLE_CNT_LOW is the min number of 300 MHz cycles squelch can be low for comwake signaling.

Offset: 0x114 | Read/Write: R/W | Reset: 0x40000000

Bit	R/W	Reset	Description
31	R/W	0	T_SATA<0>_NVOOB_RETIMED_FAREND_LOOPBACK_EN: This field when enabled puts the SATA controller in farend retimed loopback mode. 0h: RETIMED_FAREND_LOOPBACK_EN_0 (default)
30:28	R/W	4h	T_SATA<0>_NVOOB_COMMA_CNT: COMMA_CNT*16 is the number of comma characters seen for the phy_rdy to go high after going through OOB signalling. 4h: COMMA_CNT_DEFAULT (default)
27:26	R/W	0	T_SATA<0>_NVOOB_SQUELCH_FILTER_LENGTH: (SQUELCH_FILTER_LENGTH+1)*6.66ns is the amount of glitch duration that are filtered out. 0h: SQUELCH_FILTER_LENGTH_0 (default)
25:24	R/W	0	T_SATA<0>_NVOOB_SQUELCH_FILTER_MODE: This field selects squelch signal filtering mode. When FILTER_MODE_LOW is selected, high-low-high glitches are filtered out. When FILTER_MODE_HIGH is selected, low-high-low glitches are filtered out. 0h: SQUELCH_FILTER_MODE_NONE (default) 1h: SQUELCH_FILTER_MODE_LOW 2h: SQUELCH_FILTER_MODE_HIGH
23:0	R	0	Reserved

T_SATA<0>_CROSS_BAR

SATA CROSS BAR

Offset: 0x118 | Read/Write: R/W | Reset: 0x00543210

Bit	R/W	Reset	Description
31:0	R/W	00543210h	T_SATA<0>_CROSS_BAR_PHY_SEL: This register contains the phy_select - indicates how a Lane(brick) is connected to the port. NOTE: this field is reset by Cold Reset 543210h: PHY_SEL_DEFAULT (default)

T_SATA<0>_PMUCTL

SATA PHY Control Register

This register contains various bits to control the PHY.

FORCE_CORE_CLAMP -- is used to enable core clock clamp. Software can only set this bit when the controller is idle. Any downstream cycle, including a read to this register, will wake up core clock automatically, and core_act_sts will change to 0x0. Software needs no future action to un-clamp the clock. But software needs to write 1 to this bit to re-clamp the clock when the controller is idle.

Note: Wake from core clock clamp can be done by a downstream cycle which will trigger the core_wake signal to PMU; or by hotplug event which will trigger the dev_wake signal.

DEV_STS_HOLD -- # of txclk cycles to hold the dev_act_tog and dev_act_sts. CORE_STS -- current core_act_sts. DEV_STS -- current dev_act_sts.

Offset: 0x11c | Read/Write: R/W | Reset: 0xX03F00XX

Bit	R/W	Reset	Description
31:28	R	0h	T_SATA<0>_PMUCTL_RSVD_31_28: 0h: RSVD_31_28_VAL
27:24	R	0	T_SATA<0>_PMUCTL_CORE_STS: 0h: CORE_STS_DEFAULT (default)
23:16	R/W	3Fh	T_SATA<0>_PMUCTL_DEV_STS_HOLD: NOTE: this field is reset by Cold Reset 3Fh: DEV_STS_HOLD_DEFAULT (default)
15:9	R	0	Reserved
8	R/W	0	T_SATA<0>_PMUCTL_HOLD_SEND_ALIGN_DIS: NOTE: this field is reset by Cold Reset 0h: HOLD_SEND_ALIGN_DIS_NO 1h: HOLD_SEND_ALIGN_DIS_YES 0h: HOLD_SEND_ALIGN_DIS_DEFAULT (default)
7:3	R	0h	T_SATA<0>_PMUCTL_RSVD_3_7: 0h: RSVD_3_7_VAL
2	R/W	0	T_SATA<0>_PMUCTL_FORCE_CORE_CLAMP: NOTE: this field is reset by Cold Reset 1h: FORCE_CORE_CLAMP_EN 0h: FORCE_CORE_CLAMP_DIS (default)
1:0	R	0h	T_SATA<0>_PMUCTL_RSVD_0_1: 0h: RSVD_0_1_VAL

T_SATA<0>_CFG_PHY_0

Offset: 0x120 | Read/Write: R/W | Reset: 0x0000002C / 0x3228ADAC (SATA0)

Bit	R/W	Reset	Description
31:7	R	0	Reserved (SATA only)
31	R/W	0	T_SATA0_CFG_PHY_0_HOLD_RX_STAT_IDLE_IN_BIST: 0h: HOLD_RX_STAT_IDLE_IN_BIST_INIT (default)
30	R/W	0	T_SATA0_CFG_PHY_0_RESET_SREGS_EVERY_COMRESET: NOTE: this field is reset by Cold Reset 0h: RESET_SREGS_EVERY_COMRESET_INIT (default)
29	R/W	1	T_SATA0_CFG_PHY_0_ASSERT_PHYRDY_IN_NVOOB_SM: NOTE: this field is reset by Cold Reset 1h: ASSERT_PHYRDY_IN_NVOOB_SM_YES (default) 0h: ASSERT_PHYRDY_IN_NVOOB_SM_NO
28	R/W	1	T_SATA0_CFG_PHY_0_USE_STORED_COMWAKE: 1h: USE_STORED_COMWAKE_INIT (default)
27	R/W	0	T_SATA0_CFG_PHY_0_RX_STAT_IDLE: 0h: RX_STAT_IDLE_INIT (default)

Bit	R/W	Reset	Description
26	R/W	0	T_SATA0_CFG_PHY_0_ASSERT_PHYRDY_EVEN_NOT_HRRDY: NOTE: this field is reset by Cold Reset 1h: ASSERT_PHYRDY_EVEN_NOT_HRRDY_YES 0h: ASSERT_PHYRDY_EVEN_NOT_HRRDY_NO (default)
25	R/W	1	T_SATA0_CFG_PHY_0_ASSERT_PHYRDY_FOR_BIST: NOTE: this field is reset by Cold Reset 1h: ASSERT_PHYRDY_FOR_BIST_YES (default) 0h: ASSERT_PHYRDY_FOR_BIST_NO
24	R/W	0	T_SATA0_CFG_PHY_0_MASK_SQUELCH: NOTE: this field is reset by Cold Reset 1h: MASK_SQUELCH_YES 0h: MASK_SQUELCH_NO (default)
23:22	R/W	0	T_SATA0_CFG_PHY_0_RX_SLEEP_ACTIVE: NOTE: this field is reset by Cold Reset 0h: RX_SLEEP_ACTIVE_INIT (default)
21:20	R/W	2h	T_SATA0_CFG_PHY_0_RX_SLEEP_PARTIAL: NOTE: this field is reset by Cold Reset 2h: RX_SLEEP_PARTIAL_INIT (default)
19:18	R/W	2h	T_SATA0_CFG_PHY_0_RX_SLEEP_SLUMBER: NOTE: this field is reset by Cold Reset 2h: RX_SLEEP_SLUMBER_INIT (default)
17:16	R/W	0	T_SATA0_CFG_PHY_0_TX_SLEEP_ACTIVE: NOTE: this field is reset by Cold Reset 0h: TX_SLEEP_ACTIVE_INIT (default)
15:14	R/W	2h	T_SATA0_CFG_PHY_0_TX_SLEEP_PARTIAL: NOTE: this field is reset by Cold Reset 2h: TX_SLEEP_PARTIAL_INIT (default)
13:12	R/W	2h	T_SATA0_CFG_PHY_0_TX_SLEEP_SLUMBER: NOTE: this field is reset by Cold Reset 2h: TX_SLEEP_SLUMBER_INIT (default)
11	R/W	1	T_SATA0_CFG_PHY_0_USE_7BIT_ALIGN_DET_FOR_SPD: NOTE: this field is reset by Cold Reset 1h: USE_7BIT_ALIGN_DET_FOR_SPD_YES (default) 0h: USE_7BIT_ALIGN_DET_FOR_SPD_NO
10	R/W	1	T_SATA0_CFG_PHY_0_OOB_COMINIT_UNMASK: NOTE: this field is reset by Cold Reset 1h: OOB_COMINIT_UNMASK_YES (default) 0h: OOB_COMINIT_UNMASK_NO
9	R/W	0	T_SATA0_CFG_PHY_0_SEND_COMRESET_ON_WARMRESET: this bit set indicates that COMRESET will not be sent on a WARM_RESET, it will be sent only when SCTL_DET is written NOTE: this field is reset by Cold Reset 1h: SEND_COMRESET_ON_WARMRESET_YES 0h: SEND_COMRESET_ON_WARMRESET_NO (default)
8	R/W	1	T_SATA0_CFG_PHY_0_DONT_INSERT_ALIGNS_IN_BIST_L: This bit when set indicates that Aligns will not be inserted when in BIST_L mode to avoid overflow NOTE: this field is reset by Cold Reset 1h: DONT_INSERT_ALIGNS_IN_BIST_L_YES (default) 0h: DONT_INSERT_ALIGNS_IN_BIST_L_NO
7	R/W	1	T_SATA0_CFG_PHY_0_SSTS_DET_1_IN_PART_SLUMBER: This bit when set indicates that ssts_det will be made one when we go to PARTIAL/ SLUMBER, else zero NOTE: this field is reset by Cold Reset 1h: SSTS_DET_1_IN_PART_SLUMBER_YES (default) 0h: SSTS_DET_1_IN_PART_SLUMBER_NO
6	R/W	0	T_SATA<0>_CFG_PHY_0_PLL_IDDQ_OVERRIDE_VAL: NOTE: this field is reset by Cold Reset

Bit	R/W	Reset	Description
			1h: PLL_IDDQ_OVERRIDE_VAL_YES 0h: PLL_IDDQ_OVERRIDE_VAL_NO (default)
5	R/W	1	T_SATA<0>_CFG_PHY_0_PLL_IDDQ_OVERRIDE: NOTE: this field is reset by Cold Reset 1h: PLL_IDDQ_OVERRIDE_YES (default) 0h: PLL_IDDQ_OVERRIDE_NO
4:0	R/W	0Ch	T_SATA<0>_CFG_PHY_0_RBC_RESET_DELAY: RBC_RESET_DELAY is the number of tx_clk ticks (300MHz) between the deassertion of sata2phy_ch{1,2}_cdr_reset and the release of ch{1,2}_rbc_reset_. NOTE: this field is reset by Cold Reset Ch: RBC_RESET_DELAY_DEFAULT (default)

T_SATA0_CFG_PHY_POWER

This register applies to SATA0 register space only.

Offset: 0x124 | Read/Write: R/W | Reset: 0x96EA6001

Bit	R/W	Reset	Description
31:24	R/W	96h	T_SATA0_CFG_PHY_POWER_COUNT_FOR_1_US: 96h: COUNT_FOR_1_US_INIT (default)
23:10	R/W	3A98h	T_SATA0_CFG_PHY_POWER_COUNT_FOR_100_US: 3A98h: COUNT_FOR_100_US_INIT (default)
9:0	R/W	1	T_SATA0_CFG_PHY_POWER_TIME_WAIT_IN_PARTIAL: indicates the time in ms up to which the PHY SM would wait in PARTIAL before going to SLUMBER(AUTO PARTIAL TO SLUMBER feature) 1h: TIME_WAIT_IN_PARTIAL_INIT (default)

T_SATA0_CFG_PHY_POWER_1

This register applies to SATA0 register space only.

Offset: 0x128 | Read/Write: R/W | Reset: 0x0FF249F0

Bit	R/W	Reset	Description
31:20	R/W	0FFh	T_SATA0_CFG_PHY_POWER_1_COUNTER_FOR_PADS: FFh: COUNTER_FOR_PADS_INIT (default)
19:0	R/W	249F0h	T_SATA0_CFG_PHY_POWER_1_COUNT_FOR_1_MS: Indicates the number of clock cycles for one ms at 150 MHz (sata_oob_detect_clk). Used for Auto Partial to slumber feature 249F0h: COUNT_FOR_1_MS_INIT (default)

T_SATA<0>_CFG_PHY_1

Offset: 0x12c | Read/Write: R/W | Reset: 0x0732400F

Bit	R/W	Reset	Description
31:27	R	0	Reserved
26	R/W	1	T_SATA<0>_CFG_PHY_1_DONT_CHK_PHY_RESET: NOTE: this field is reset by Cold Reset 1h: DONT_CHK_PHY_RESET_INIT (default)
25	R/W	1	T_SATA<0>_CFG_PHY_1_COMWAKE_GLOBAL: NOTE: this field is reset by Cold Reset 1h: COMWAKE_GLOBAL_INIT (default)
24	R/W	1	T_SATA<0>_CFG_PHY_1_PLL_PD_NO_CMDS:

Bit	R/W	Reset	Description
			1h: PLL_PD_NO_CMDS_INIT (default)
23	R/W	0	T_SATA<0>_CFG_PHY_1_PADS_IDDQ_EN: NOTE: this field is reset by Cold Reset 0h: PADS_IDDQ_EN_INIT (default)
22	R/W	0	T_SATA<0>_CFG_PHY_1_PAD_PLL_IDDQ_EN: NOTE: this field is reset by Cold Reset 0h: PAD_PLL_IDDQ_EN_INIT (default)
21	R/W	1	T_SATA<0>_CFG_PHY_1_SEND_OOB_DATA_IN_LOW_POWER: NOTE: this field is reset by Cold Reset 1h: SEND_OOB_DATA_IN_LOW_POWER_INIT (default)
20	R/W	1	T_SATA<0>_CFG_PHY_1_NO_OVERRIDE_STAT_IDLE_PHYRDY: NOTE: this field is reset by Cold Reset 1h: NO_OVERRIDE_STAT_IDLE_PHYRDY_INIT (default)
19:16	R/W	2h	T_SATA<0>_CFG_PHY_1_NUMBER_OF_COMMA_WINDOWS:
15:4	R/W	400h	T_SATA<0>_CFG_PHY_1_COUNT_FOR_COMMA_WAIT: NOTE: this field is reset by Cold Reset 400h: COUNT_FOR_COMMA_WAIT_INIT (default)
3	R/W	1	T_SATA<0>_CFG_PHY_1_DONT_USE_COMMA_FOR_PHYRDY_LOW: NOTE: this field is reset by Cold Reset 1h: DONT_USE_COMMA_FOR_PHYRDY_LOW_INIT (default)
2	R/W	1	T_SATA<0>_CFG_PHY_1_EN_ASYNC_REC_ARC_IN_HRRDY: NOTE: this field is reset by Cold Reset 1h: EN_ASYNC_REC_ARC_IN_HRRDY_INIT (default)
1	R/W	1	T_SATA<0>_CFG_PHY_1_HOLD_RBC_RESET_IN_BIST: 1h: HOLD_RBC_RESET_IN_BIST_INIT (default)
0	R/W	1	T_SATA<0>_CFG_PHY_1_ASSERT_PHYRDY_FOR_ALL_BIST: 1h: ASSERT_PHYRDY_FOR_ALL_BIST_INIT (default)

T_SATA<0>_CFG2NVOOB_1

Offset: 0x130 | Read/Write: R/W | Reset: 0x01802220

Bit	R/W	Reset	Description
31:27	R	0	Reserved
26:18	R/W	060h	T_SATA<0>_CFG2NVOOB_1_COMINIT_IDLE_CNT_HIGH: Squelch can be high for a maximum of COMINIT_IDLE_CNT_HIGH cycles at 216MHz for COMINIT signalling. NOTE: this field is reset by Cold Reset 60h: COMINIT_IDLE_CNT_HIGH_DEFAULT (default)
17:9	R/W	011h	T_SATA<0>_CFG2NVOOB_1_ACTIVE_CNT_LOW: Squelch can be high for a mininum of ACTIVE_CNT_LOW cycles at 216MHz for OOB signalling. NOTE: this field is reset by Cold Reset 11h: ACTIVE_CNT_LOW_DEFAULT (default)
8:0	R/W	020h	T_SATA<0>_CFG2NVOOB_1_ACTIVE_CNT_HIGH: Squelch can be high for a maximum of ACTIVE_CNT_HIGH cycles at 216MHz for OOB signalling. NOTE: this field is reset by Cold Reset 20h: ACTIVE_CNT_HIGH_DEFAULT (default)

T_SATA<0>_CFG2NVOOB_2

Offset: 0x134 | Read/Write: R/W | Reset: 0x00444C38

Bit	R/W	Reset	Description

Bit	R/W	Reset	Description
31:27	R	0	Reserved
26:18	R/W	011h	T_SATA<0>_CFG2NVOOB_2_COMWAKE_IDLE_CNT_LOW: Squelch can be high for a minimum of 2*COMWAKE_IDLE_CNT_LOW cycles at 216MHz for COMWAKE signalling. NOTE: this field is reset by Cold Reset 11h: COMWAKE_IDLE_CNT_LOW_DEFAULT (default)
17:9	R/W	026h	T_SATA<0>_CFG2NVOOB_2_COMWAKE_IDLE_CNT_HIGH: Squelch can be high for a maximum of 32 + 2*COMWAKE_IDLE_CNT_HIGH cycles at 216MHz for COMWAKE signalling. NOTE: this field is reset by Cold Reset 26h: COMWAKE_IDLE_CNT_HIGH_DEFAULT (default)
8:0	R/W	038h	T_SATA<0>_CFG2NVOOB_2_COMINIT_IDLE_CNT_LOW: Squelch can be high for a minimum of 50 + 4*COMINIT_IDLE_CNT_LOW cycles at 216MHz for COMINIT signalling. NOTE: this field is reset by Cold Reset 38h: COMINIT_IDLE_CNT_LOW_DEFAULT (default)

T_SATA<0>_CFG_PHY_ACTIVE

Offset: 0x138 | Read/Write: R/W | Reset: 0x0001705C

Bit	R/W	Reset	Description
31:10	R/W	00005Ch	T_SATA<0>_CFG_PHY_ACTIVE_FROM_SLUMBER: Decides the time to be in SLUMEBR state after we receive a COMWAKE from the device or the Link Layer lowers the slumber flag. To give time for the pads to wakeup before sending the COMWAKE. This is for txclk of 300MHZ, will be multiplied by two in the OOB state-machine for 600MHz Changing the default value of SLUMBER to 0x5C since Partial and Slumber wakeup times are the same. NOTE: this field is reset by Cold Reset 3AA8h: FROM_SLUMBER_DEFAULT_1 5Ch: FROM_SLUMBER_DEFAULT (default)
9:0	R/W	05Ch	T_SATA<0>_CFG_PHY_ACTIVE_FROM_PARTIAL: Decides the time to be in PARTIAL state after we receive a COMWAKE from the device or the Link Layer lowers the partial flag. To give time for the pads to wakeup before sending the COMWAKE. This is for txclk of 300MHZ, will be multiplied by two in the OOB state-machine for 600MHz NOTE: this field is reset by Cold Reset 5Ch: FROM_PARTIAL_DEFAULT (default)

T_SATA<0>_FIFO

Offset: 0x170 | Read/Write: R/W | Reset: 0x00086000

Bit	R/W	Reset	Description
31:20	R	0	Reserved
19:16	R/W	8h	T_SATA<0>_FIFO_P2L_FIFO_DEPTH: This controls effective depth of p2l FIFO. Smaller is better for performance but we can't do power optimization (clamping fpci_clk) with smaller depth FIFOs. NOTE: this field is reset by Cold Reset 8h: P2L_FIFO_DEPTH_DEFAULT (default)
15:12	R/W	6h	T_SATA<0>_FIFO_L2P_FIFO_DEPTH: This controls effective depth of l2p FIFO. Smaller is better for performance but we can't do power optimization (clamping fpci_clk) with smaller depth FIFOs. NOTE: this field is reset by Cold Reset 6h: L2P_FIFO_DEPTH_DEFAULT (default)
11:0	R	0	Reserved

T_SATA<0>_CFG_LINK_0

Offset: 0x174 | Read/Write: R/W | Reset: 0xXX00009A (SATA) / 0xXXDD239A (SATA0)

Bit	R/W	Reset	Description
31:25	R	0h	T_SATA<0>_CFG_LINK_0_RSVD: 0h: RSVD_VAL
24:8	0	R	Reserved (SATA only)
24	R/W	0	T_SATA0_CFG_LINK_0_USE_POSEDGE_SCTL_DET: NOTE: this field is reset by Cold Reset 0h: USE_POSEDGE_SCTL_DET_INIT (default)
23	R/W	1	T_SATA0_CFG_LINK_0_USE_DET_OR_PSM2LL_RESET: NOTE: this field is reset by Cold Reset 1h: USE_DET_OR_PSM2LL_RESET_INIT (default)
22	R/W	1	T_SATA0_CFG_LINK_0_HARDCODE_DISPARITY_ON_WAKE: NOTE: this field is reset by Cold Reset 1h: HARDCODE_DISPARITY_ON_WAKE_YES (default) 0h: HARDCODE_DISPARITY_ON_WAKE_NO
21	R/W	0	T_SATA0_CFG_LINK_0_USE_IS_PRIM_FOR_BIST: 0h: USE_IS_PRIM_FOR_BIST_INIT (default)
20	R/W	1	T_SATA0_CFG_LINK_0_WAIT_FOR_PSM_FOR_PMOFF: NOTE: this field is reset by Cold Reset 1h: WAIT_FOR_PSM_FOR_PMOFF_INIT (default)
19	R/W	1	T_SATA0_CFG_LINK_0_USE_AHCI_MODE_FOR_COMRESET: NOTE: this field is reset by Cold Reset 1h: USE_AHCI_MODE_FOR_COMRESET_YES (default)
18	R/W	1	T_SATA0_CFG_LINK_0_GOTO_WAKE_UP4: NOTE: this field is reset by Cold Reset 1h: GOTO_WAKE_UP4_YES (default) 0h: GOTO_WAKE_UP4_NO
17	R/W	0	T_SATA0_CFG_LINK_0_DONT_CHK_PHYRDY_IN_NO_COMM: NOTE: this field is reset by Cold Reset 1h: DONT_CHK_PHYRDY_IN_NO_COMM_YES 0h: DONT_CHK_PHYRDY_IN_NO_COMM_NO (default)
16	R/W	1	T_SATA0_CFG_LINK_0_SEND_NEG_RD_ALIGNS: NOTE: this field is reset by Cold Reset 1h: SEND_NEG_RD_ALIGNS_YES (default) 0h: SEND_NEG_RD_ALIGNS_NO
15	R/W	0	T_SATA0_CFG_LINK_0_OLD_BEHAVIOUR_SEND_ALIGNS: NOTE: this field is reset by Cold Reset 1h: OLD_BEHAVIOUR_SEND_ALIGNS_YES 0h: OLD_BEHAVIOUR_SEND_ALIGNS_NO (default)
14:10	R/W	08h	T_SATA0_CFG_LINK_0_PM_OFF_COUNT: Indicates the number of PMREQ send to the device on Slumber/ PARTIAL NOTE: this field is reset by Cold Reset 8h: PM_OFF_COUNT_DEFAULT (default)
9	R/W	1	T_SATA0_CFG_LINK_0_SEND_RAW_DATA_IN_BIST_L: Default to zero, when set sends raw data in BIST_L mode without encoding/ decoding NOTE: this field is reset by Cold Reset 1h: SEND_RAW_DATA_IN_BIST_L_YES (default) 0h: SEND_RAW_DATA_IN_BIST_L_NO
8	R/W	1	T_SATA0_CFG_LINK_0_BE_IN_BIST_ON_PHYRDY_LOW: The default value of this bit is one, Host should come out of BIST only on COMINIT/ COMRESET, default to that NOTE: this field is reset by Cold Reset 1h: BE_IN_BIST_ON_PHYRDY_LOW_YES (default) 0h: BE_IN_BIST_ON_PHYRDY_LOW_NO

Bit	R/W	Reset	Description
7	R/W	1	T_SATA<0>_CFG_LINK_0_GOTO_SEND_SYNC_P: This bit set sends Link SM to L_SEND_SYNC_P in BIST NOTE: this field is reset by Cold Reset 0h: GOTO_SEND_SYNC_P_NO 1h: GOTO_SEND_SYNC_P_YES (default)
6	R/W	0	T_SATA<0>_CFG_LINK_0_AUTO_REPEAT_PRIMS: NOTE: this field is reset by Cold Reset 0h: AUTO_REPEAT_PRIMS_NO 1h: AUTO_REPEAT_PRIMS_YES 0h: AUTO_REPEAT_PRIMS_INIT (default)
5	R/W	0	T_SATA<0>_CFG_LINK_0_DEBOUNCE_PHYRDY_PERIOD: When set to 1, the hysteresis period is let to LONG mode. A setting of zero is SHORT mode. NOTE: this field is reset by Cold Reset 1h: DEBOUNCE_PHYRDY_PERIOD_LONG 0h: DEBOUNCE_PHYRDY_PERIOD_SHORT 0h: DEBOUNCE_PHYRDY_PERIOD_INIT (default)
4	R/W	1	T_SATA<0>_CFG_LINK_0_DEBOUNCE_PHYRDY: This bit and next bit controls phyrdy debounce behavior. By default phyrdy is debounced. Once bit this is set, bit 5 selects the period of hysteresis. NOTE: this field is reset by Cold Reset 1h: DEBOUNCE_PHYRDY_YES 0h: DEBOUNCE_PHYRDY_NO 1h: DEBOUNCE_PHYRDY_INIT (default)
3	R/W	1	T_SATA<0>_CFG_LINK_0_DELAY_HOTPLUG_INTR: This bit is used to control generation of hotplug interrupts. If this bit is set to 1, then an interrupt is generated when we get PHYRDY. If this bit is set to 0, then an interrupt is generated as soon as cominit is received. NOTE: this field is reset by Cold Reset 1h: DELAY_HOTPLUG_INTR_YES (default) 0h: DELAY_HOTPLUG_INTR_NO
2	R/W	0	T_SATA<0>_CFG_LINK_0_SET_BSY_BIT_MTHD: This controls when 8'h80 is loaded into shadow status reg. If the bit is 0 then 8'h80 is loaded when cominit is seen otherwise its loaded when phyrdy is seen. NOTE: this field is reset by Cold Reset 1h: SET_BSY_BIT_MTHD_PHYRDY 0h: SET_BSY_BIT_MTHD_COMINIT (default)
1:0	R/W	2h	T_SATA<0>_CFG_LINK_0_LED_MIN_ON_TIME: Used to know the activity based on LED blinking NOTE: this field is reset by Cold Reset 0h: LED_MIN_ON_TIME_OFF 1h: LED_MIN_ON_TIME_20MS 2h: LED_MIN_ON_TIME_40MS (default) 3h: LED_MIN_ON_TIME_80MS

T_SATA<0>_CFG_LINK_1

Offset: 0x178 | Read/Write: R/W | Reset: 0x02000200

Bit	R/W	Reset	Description
31:16	R/W	0200h	T_SATA<0>_CFG_LINK_1_GEN3_DWRD_WAIT_CNT: This field is the number of generation3 ALIGN dwords the controller sends to drive while waiting for gen3 align from drive, before it gives up and switches to gen2 mode of operation. It should be set to a value smaller than 2*2048. It should also be divisible by 2. NOTE: this field is reset by Cold Reset 200h: GEN3_DWRD_WAIT_CNT_INIT (default)
15:0	R/W	0200h	T_SATA<0>_CFG_LINK_1_GEN2_DWRD_WAIT_CNT: This field is the number of generation2 ALIGN dwords the controller sends to drive while waiting for gen2 align from drive, before it gives up and switches to gen1 mode of operation. It should be set to a value smaller than 2*2048. It should also be divisible by 2. NOTE: this field is reset by Cold Reset

Bit	R/W	Reset	Description
			200h: GEN2_DWRD_WAIT_CNT_INIT (default)

T_SATA<0>_CFG_LINK_2

Offset: 0x17c | Read/Write: R/W | Reset: 0x00007080

Bit	R/W	Reset	Description
31:16	R	0	Reserved
15:12	R/W	7h	T_SATA<0>_CFG_LINK_2_PHYRDY_USE_ITH_BIT: NOTE: this field is reset by Cold Reset 7h: PHYRDY_USE_ITH_BIT_INIT (default)
11	R/W	0	T_SATA<0>_CFG_LINK_2_USE_BIT_FOR_DEBOUNCE: NOTE: this field is reset by Cold Reset 0h: USE_BIT_FOR_DEBOUNCE_INIT (default)
10:0	R/W	080h	T_SATA<0>_CFG_LINK_2_PHYRDY_DBNC_MUX: We have a debounced version of PHYRDY which has two options 10 ms and 10 µs (for a clock of around 100MHz - devclk) NOTE: this field is reset by Cold Reset 80h: PHYRDY_DBNC_MUX_INIT (default)

T_SATA<0>_CFG_TRANS_0

Offset: 0x1d0 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:2	R	0h	T_SATA<0>_CFG_TRANS_0_RSVD: 0h: RSVD_VAL
1	R/W	1	T_SATA<0>_CFG_TRANS_0_USE_RISE_EDGE_STATUS_RESET: NOTE: this field is reset by Cold Reset 1h: USE_RISE_EDGE_STATUS_RESET_YES (default) 0h: USE_RISE_EDGE_STATUS_RESET_NO
0	R/W	0	T_SATA<0>_CFG_TRANS_0_F2I_FIFO_FLUSH_FIX: NOTE: this field is reset by Cold Reset 1h: F2I_FIFO_FLUSH_FIX_YES 0h: F2I_FIFO_FLUSH_FIX_NO (default)

T_SATA0_ALPM_CTRL

ALPM Controls

This register applies to SATA0 register space only.

Offset: 0x238 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:16	R/W	0	T_SATA0_ALPM_CTRL_THRESHOLD: This field is used to indicate to HBA as to how much time it should wait before going to Partial or Slumber power down states. It is used to load the 16-bit start value of a down-counter in RTL. Default value zero indicates that if PxCI and PxSACT registers are cleared to zero (implies that if there are no commands outstanding) the HBA SM will transition to initiate power down. NOTE: this field is reset by Cold Reset 0h: THRESHOLD_DEFAULT (default)
15:0	R	0	Reserved

T_SATA0_FBS_CONFIG_0

This register implements settings for FIS based switching implementation.

This register applies to SATA0 register space only.

Offset: 0x23c | Read/Write: R/W | Reset: 0xF0F0DFF9

Bit	R/W	Reset	Description
31:16	R/W	F0F0h	T_SATA0_FBS_CONFIG_0_CTL_03_RSVD: F0F0h: CTL_03_RSVD_INIT (default)
15:14	R/W	3h	T_SATA0_FBS_CONFIG_0_PRD_PROCESS_FIX: 3h: PRD_PROCESS_FIX_INIT (default)
13	R/W	0	T_SATA0_FBS_CONFIG_0_POST_1ST_REGFIS: 0h: POST_1ST_REGFIS_INIT (default)
12	R/W	1	T_SATA0_FBS_CONFIG_0_CTL_02: 1h: CTL_02_INIT (default)
11:8	R/W	Fh	T_SATA0_FBS_CONFIG_0_BKDR_ADO: Fh: BKDR_ADO_INIT (default)
7:4	R/W	Fh	T_SATA0_FBS_CONFIG_0_CONTROL_PORT: Fh: CONTROL_PORT_INIT (default)
3	R/W	1	T_SATA0_FBS_CONFIG_0_CONTROL_PORT_HAS_PRIORITY: 1h: CONTROL_PORT_HAS_PRIORITY_INIT (default)
2	R/W	0	T_SATA0_FBS_CONFIG_0_CTL_02_RSVD: 0h: CTL_02_RSVD_INIT (default)
1	R/W	0	T_SATA0_FBS_CONFIG_0_CTL_01: 0h: CTL_01_INIT (default)
0	R/W	1	T_SATA0_FBS_CONFIG_0_CTL_00: 1h: CTL_00_INIT (default)

T_SATA0_CHX_FBS_CONFIG_1

This register implements settings for FIS based switching implementation.

This register applies to SATA0 register space only.

Offset: 0x240 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:1	R	0h	T_SATA0_CHX_FBS_CONFIG_1_RSVD_1_31: 0h: RSVD_1_31_VAL
0	R/W	0	T_SATA0_CHX_FBS_CONFIG_1_PORT_BKDR_FBSCP: NOTE: this field is reset by Cold Reset 0h: PORT_BKDR_FBSCP_INIT (default)

T_SATA0_AHCI_HBA_CAP_BKDR

This register will have a backdoor field for every bit described in the AHCI 1.2 specification.

This register applies to SATA0 register space only.

Offset: 0x300 | Read/Write: R/W | Reset: 0xE620FF01

Bit	R/W	Reset	Description
31	R/W	1	T_SATA0_AHCI_HBA_CAP_BKDR_S64A: 1h: S64A_TRUE (default)

Bit	R/W	Reset	Description
			0h: S64A_FALSE
30	R/W	1	T_SATA0_AHCI_HBA_CAP_BKDR_SNCQ: 1h: SNCQ_TRUE (default) 0h: SNCQ_FALSE
29	R/W	1	T_SATA0_AHCI_HBA_CAP_BKDR_SSNTF: 1h: SSNTF_TRUE (default) 0h: SSNTF_FALSE
28	R/W	0	T_SATA0_AHCI_HBA_CAP_BKDR_SMPS: 1h: SMPS_TRUE 0h: SMPS_FALSE (default)
27	R/W	0	T_SATA0_AHCI_HBA_CAP_BKDR_SUPP_STG_SPUP: Backdoor field to advertise staggered spin up. BIOS has to write this we are going to support staggered spin up. 1h: SUPP_STG_SPUP_SET 0h: SUPP_STG_SPUP_CLEAR (default)
26	R/W	1	T_SATA0_AHCI_HBA_CAP_BKDR_SALP: 1h: SALP_TRUE (default) 0h: SALP_FALSE
25	R/W	1	T_SATA0_AHCI_HBA_CAP_BKDR_SAL: 1h: SAL_TRUE (default) 0h: SAL_FALSE
24	R/W	0	T_SATA0_AHCI_HBA_CAP_BKDR_SUPP_CLO: 1h: SUPP_CLO_TRUE 0h: SUPP_CLO_FALSE (default)
23:20	R/W	2h	T_SATA0_AHCI_HBA_CAP_BKDR_INTF_SPD_SUPP: 0h: INTF_SPD_SUPP_RSVD 1h: INTF_SPD_SUPP_GEN1 2h: INTF_SPD_SUPP_GEN1_2 (default) 3h: INTF_SPD_SUPP_GEN3
19	R/W	0	T_SATA0_AHCI_HBA_CAP_BKDR_SUPP_NONZERO_OFFSET: 1h: SUPP_NONZERO_OFFSET_TRUE 0h: SUPP_NONZERO_OFFSET_FALSE (default)
18	R/W	0	T_SATA0_AHCI_HBA_CAP_BKDR_SUPP_AHCI_ONLY: 1h: SUPP_AHCI_ONLY_TRUE 0h: SUPP_AHCI_ONLY_FALSE (default)
17	R/W	0	T_SATA0_AHCI_HBA_CAP_BKDR_SUPP_PM: 1h: SUPP_PM_TRUE 0h: SUPP_PM_FALSE (default)
16	R/W	0	T_SATA0_AHCI_HBA_CAP_BKDR_FIS_SWITCHING: 1h: FIS_SWITCHING_TRUE 0h: FIS_SWITCHING_FALSE (default)
15	R/W	1	T_SATA0_AHCI_HBA_CAP_BKDR_PIO_MULT_DRQ_BLK: 1h: PIO_MULT_DRQ_BLK_SUPP (default) 0h: PIO_MULT_DRQ_BLK_NOT_SUPP
14	R/W	1	T_SATA0_AHCI_HBA_CAP_BKDR_SLUMBER_ST_CAP: 1h: SLUMBER_ST_CAP_TRUE (default) 0h: SLUMBER_ST_CAP_FALSE
13	R/W	1	T_SATA0_AHCI_HBA_CAP_BKDR_PARTIAL_ST_CAP: 1h: PARTIAL_ST_CAP_TRUE (default) 0h: PARTIAL_ST_CAP_FALSE
12:8	R/W	1Fh	T_SATA0_AHCI_HBA_CAP_BKDR_NUM_CMD_SLOTS: 0h: NUM_CMD_SLOTS_1 7h: NUM_CMD_SLOTS_8 Fh: NUM_CMD_SLOTS_16 1Fh: NUM_CMD_SLOTS_32 (default)

Bit	R/W	Reset	Description
7	R/W	0	T_SATA0_AHCI_HBA_CAP_BKDR_CMD_CMPL_COALESING: 1h: CMD_CMPL_COALESING_TRUE 0h: CMD_CMPL_COALESING_FALSE (default)
6	R/W	0	T_SATA0_AHCI_HBA_CAP_BKDR_ENCL_MGMT_SUPP: 1h: ENCL_MGMT_SUPP_TRUE 0h: ENCL_MGMT_SUPP_FALSE (default)
5	R/W	0	T_SATA0_AHCI_HBA_CAP_BKDR_EXT_SATA: This is a backdoor access for enabling external sata. 1h: EXT_SATA_SUPPORTED 0h: EXT_SATA_NOT_SUPPORTED (default)
4:0	R/W	1	T_SATA0_AHCI_HBA_CAP_BKDR_NUM_PORTS: 0h: NUM_PORTS_1 1h: NUM_PORTS_2 2h: NUM_PORTS_3 3h: NUM_PORTS_4 1h: NUM_PORTS_DEFUALT (default)

T_SATA0_AHCI_HBA_HOLD_GEN

This register applies to SATA0 register space only.

Offset: 0x304 | Read/Write: R/W | Reset: 0x19080026

Bit	R/W	Reset	Description
31:24	R/W	19h	T_SATA0_AHCI_HBA_HOLD_GEN_SHARED_DFIFO_AVAIL: This field is programmed with a variable water mark for generation of HOLD to drive depending on depth of shared DATA FIFO. This field indicates the number of lines of the FIFO and not the size in bytes/words. NOTE: this field is reset by Cold Reset 19h: SHARED_DFIFO_AVAIL_INIT (default)
23:8	R/W	0800h	T_SATA0_AHCI_HBA_HOLD_GEN_PRD_SIZE: This field indicates the minimum value of PRDBC for the last PRD cached below which Port should send a HOLD to drive. NOTE: this field is reset by Cold Reset 800h: PRD_SIZE_INIT (default)
7:0	R/W	26h	T_SATA0_AHCI_HBA_HOLD_GEN_T2P_FIFO_AVAIL: This field is programmed with a variable watermark for generation of HOLD to drive depending on the depth of T2P FIFO. This field indicates the number of lines of the FIFO and not the size in bytes/words. NOTE: this field is reset by Cold Reset 26h: T2P_FIFO_AVAIL_INIT (default) 2Eh: T2P_FIFO_AVAIL_MAX

T_SATA0_AHCI_HBA_CTL_0

This register applies to SATA0 register space only.

Offset: 0x30c | Read/Write: R/W | Reset: 0x7C070014

Bit	R/W	Reset	Description
31	R/W	0	T_SATA0_AHCI_HBA_CTL_0_USE_PXCMD_ICC_IF_LINK_IN_IDLE: 1h: USE_PXCMD_ICC_IF_LINK_IN_IDLE_YES 0h: USE_PXCMD_ICC_IF_LINK_IN_IDLE_NO (default)
30	R/W	1	T_SATA0_AHCI_HBA_CTL_0_USE_AHCI_MODE_IN_FIS_PROCESS: 1h: USE_AHCI_MODE_IN_FIS_PROCESS_YES (default) 0h: USE_AHCI_MODE_IN_FIS_PROCESS_NO
29:28	R/W	3h	T_SATA0_AHCI_HBA_CTL_0_BKDR_VS_MINOR_9_8: 3h: BKDR_VS_MINOR_9_8_DEFAULT (default)

Bit	R/W	Reset	Description
27	R/W	1	T_SATA0_AHCI_HBA_CTL_0_TRANSPRT_IGNORE_DMAT: 1h: TRANSPRT_IGNORE_DMAT_TRUE (default) 0h: TRANSPRT_IGNORE_DMAT_FALSE
26	R/W	1	T_SATA0_AHCI_HBA_CTL_0_PSM2LL_DENY_PMREQ: 1h: PSM2LL_DENY_PMREQ_TRUE (default) 0h: PSM2LL_DENY_PMREQ_FALSE
25	R/W	0	T_SATA0_AHCI_HBA_CTL_0_SUD_OLD_LOGIC: NOTE: this field is reset by Cold Reset 1h: SUD_OLD_LOGIC_TRUE 0h: SUD_OLD_LOGIC_FALSE (default)
24	R/W	0	T_SATA0_AHCI_HBA_CTL_0_RSVD_24: NOTE: this field is reset by Cold Reset 0h: RSVD_24_ZERO (default)
23:20	R/W	0	T_SATA0_AHCI_HBA_CTL_0_ACCEL_SYNC_ESC_TIMEOUT: 0h: ACCEL_SYNC_ESC_TIMEOUT_DEFAULT (default)
19	R/W	0	T_SATA0_AHCI_HBA_CTL_0_RST_AE_WITHOUT_COMRESET: NOTE: this field is reset by Cold Reset 0h: RST_AE_WITHOUT_COMRESET_DEFAULT (default)
18	R/W	1	T_SATA0_AHCI_HBA_CTL_0_RST_AE_ON_HR: NOTE: this field is reset by Cold Reset 1h: RST_AE_ON_HR_DEFAULT (default)
17	R/W	1	T_SATA0_AHCI_HBA_CTL_0_HBFS_ON_PRD_ADR_EXCEED_40B: 1h: HBFS_ON_PRD_ADR_EXCEED_40B_INIT (default)
16	R/W	1	T_SATA0_AHCI_HBA_CTL_0_HBFS_ON_BME_RESET_WHILE_ACTIVE: 1h: HBFS_ON_BME_RESET_WHILE_ACTIVE_DEFAULT (default)
15:8	R	0	Reserved
7	R/W	0	T_SATA0_AHCI_HBA_CTL_0_ACCEL_WAIT_FOR_BSY0: 1h: ACCEL_WAIT_FOR_BSY0_ENABLE 0h: ACCEL_WAIT_FOR_BSY0_DISABLE (default)
6	R/W	0	T_SATA0_AHCI_HBA_CTL_0_NO_DHRS_AT_UNLOAD: 1h: NO_DHRS_AT_UNLOAD_ENABLE 0h: NO_DHRS_AT_UNLOAD_DISABLE (default)
5	R/W	0	T_SATA0_AHCI_HBA_CTL_0_MPIS_SET_AT_ACCEL: 1h: MPIS_SET_AT_ACCEL_ENABLE 0h: MPIS_SET_AT_ACCEL_DISABLE (default)
4	R/W	1	T_SATA0_AHCI_HBA_CTL_0_DIS_CCC_TIMEOUT_INT_AT_ZERO_OUTSTD_CMD: 1h: DIS_CCC_TIMEOUT_INT_AT_ZERO_OUTSTD_CMD_SET (default) 0h: DIS_CCC_TIMEOUT_INT_AT_ZERO_OUTSTD_CMD_CLEAR
3	R/W	0	T_SATA0_AHCI_HBA_CTL_0_BKDR_PRDBC_UPDATE: AHCI spec says that we need to update PRDBC before clearing PXCI for a non-native command. And optionally may update after each data FIS completion. 1h: BKDR_PRDBC_UPDATE_AT_EACH_DATA_FIS 0h: BKDR_PRDBC_UPDATE_AT_REG_FIS (default)
2	R/W	1	T_SATA0_AHCI_HBA_CTL_0_PORTX_NONNCQ_TX_RX_UNDERFLOW: 0h: PORTX_NONNCQ_TX_RX_UNDERFLOW_DISABLE 1h: PORTX_NONNCQ_TX_RX_UNDERFLOW_ENABLE (default)
1	R/W	0	T_SATA0_AHCI_HBA_CTL_0_PORTX_PRD_UNDERFLOW_IFS_ATAPI: 0h: PORTX_PRD_UNDERFLOW_IFS_ATAPI_DISABLE (default) 1h: PORTX_PRD_UNDERFLOW_IFS_ATAPI_ENABLE
0	R/W	0	T_SATA0_AHCI_HBA_CTL_0_PORTX_OFS_IN_ATAPI: 0h: PORTX_OFS_IN_ATAPI_DISABLE (default) 1h: PORTX_OFS_IN_ATAPI_ENABLE

T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL

This register has overrides for various BIST modes available according to the specification

This register applies to SATA0 register space only.

Offset: 0x318 | Read/Write: R/W | Reset: 0x00000600

Bit	R/W	Reset	Description
31:24	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_PORTS: 0h: PORTS_DEFAULT (default)
23:11	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_RSVD: 0h: RSVD_DEFAULT (default)
10	R/W	1	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_DISABLE_ASYNC_RECOVERY: 1h: DISABLE_ASYNC_RECOVERY_YES (default)
9	R/W	1	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_USE_BIST_F_MODE: 1h: USE_BIST_F_MODE_YES (default)
8	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_F: 0h: F_DEFAULT (default)
7	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_BIST_DWORD: 0h: BIST_DWORD_DEFAULT (default)
6	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_BIST_L_RDY: 0h: BIST_L_RDY_DEFAULT (default)
5	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_BIST_T_RDY: 0h: BIST_T_RDY_DEFAULT (default)
4	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_P: 0h: P_DEFAULT (default)
3	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_S: 0h: S_DEFAULT (default)
2	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_T: 0h: T_DEFAULT (default)
1	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_A: 0h: A_DEFAULT (default)
0	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_CLR: 0h: CLR_DEFAULT (default)

T_SATA0_AHCI_HBA_BIST_DWORD

This register has the Dword that is used in the second and third Dword of BIST FIS.

This register applies to SATA0 register space only.

Offset: 0x31c | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	T_SATA0_AHCI_HBA_BIST_DWORD_DATA: 0h: DATA_DEFAULT (default)

T_SATA0_AHCI_HBA_SPARE_1

This register applies to SATA0 register space only.

Offset: 0x320 | Read/Write: R/W | Reset: 0x00030D40

Bit	R/W	Reset	Description

Bit	R/W	Reset	Description
31:25	R/W	0	T_SATA0_AHCI_HBA_SPARE_1_RSVD: NOTE: this field is reset by Cold Reset 0h: RSVD_ZERO (default)
24	R	0	Reserved
23:0	R/W	030D40h	T_SATA0_AHCI_HBA_SPARE_1_MS_TIMER_CNT: This gives the count of fpci_clk for counting to millisecond. NOTE: this field is reset by Cold Reset 30D40h: MS_TIMER_CNT_DEFAULT (default) 30D40h: MS_TIMER_CNT_200MHZ 3D090h: MS_TIMER_CNT_250MHZ 493E0h: MS_TIMER_CNT_300MHZ 55730h: MS_TIMER_CNT_350MHZ 61A80h: MS_TIMER_CNT_400MHZ 7A120h: MS_TIMER_CNT_500MHZ

T_SATA0_AHCI_HBA_SPARE_2

This register applies to SATA0 register space only.

Offset: 0x324 | Read/Write: R/W | Reset: 0x00009000

Bit	R/W	Reset	Description
31:16	R/W	0	T_SATA0_AHCI_HBA_SPARE_2_RSVD: 0h: RSVD_ZERO (default)
15	R/W	1	T_SATA0_AHCI_HBA_SPARE_2_LINK_IN_PROCESS_OF_LOW_POWER: 1h: LINK_IN_PROCESS_OF_LOW_POWER_DEFAULT (default)
14	R/W	0	T_SATA0_AHCI_HBA_SPARE_2_DONT_USE_LD_IN_COUNTER: 0h: DONT_USE_LD_IN_COUNTER_INIT (default)
13	R/W	0	T_SATA0_AHCI_HBA_SPARE_2_SW_COMWAKE_CONTINUOUS: 1h: SW_COMWAKE_CONTINUOUS_ENABLE 0h: SW_COMWAKE_CONTINUOUS_DISABLE (default)
12	R/W	1	T_SATA0_AHCI_HBA_SPARE_2_ASYNC_RECOVERY_ENABLE: 1h: ASYNC_RECOVERY_ENABLE_TURE (default) 0h: ASYNC_RECOVERY_ENABLE_FALSE
11:8	R/W	0	T_SATA0_AHCI_HBA_SPARE_2_ASYNC_RECOVERY_TIMER_SEL: 0h: ASYNC_RECOVERY_TIMER_SEL_DEFAULT (default)
7:0	R	0	Reserved

T_SATA0_AHCI_HBA_DYN_CLK_CLAMP

DYN_CLK_CLAMP_TIMER is used to wait for some time after the transactions on FPCI and CSM are done to shut down the devclk.

This register applies to SATA0 register space only.

Offset: 0x328 | Read/Write: R/W | Reset: 0x02000207

Bit	R/W	Reset	Description
31:24	R/W	02h	T_SATA0_AHCI_HBA_DYN_CLK_CLAMP_TIMER_PSM: 2h: TIMER_PSM_INIT (default)
23:16	R/W	0	T_SATA0_AHCI_HBA_DYN_CLK_CLAMP_SPARE_1: 0h: SPARE_1_INIT (default)
15:8	R/W	02h	T_SATA0_AHCI_HBA_DYN_CLK_CLAMP_TIMER: 2h: TIMER_DEFAULT (default)

Bit	R/W	Reset	Description
7:3	R/W	0	T_SATA0_AHCI_HBA_DYN_CLK_CLAMP_SPARE_0: 0h: SPARE_0_INIT (default)
2:0	R/W	7h	T_SATA0_AHCI_HBA_DYN_CLK_CLAMP_SPARE_0_2: 7h: SPARE_0_2_INIT (default)

T_SATA0_AHCI_CFG_ERR_CTRL

This register implement settings for controlling behavior of AHCI SM during error conditions.

This register applies to SATA0 register space only.

Offset: 0x32c | Read/Write: R/W | Reset: 0x0000003F

Bit	R/W	Reset	Description
31:6	R	0	Reserved
5	R/W	1	T_SATA0_AHCI_CFG_ERR_CTRL_ENABLE_PXIS_IFS_DATA_FIS_CRC_ERROR: 1h: ENABLE_PXIS_IFS_DATA_FIS_CRC_ERROR_INIT (default)
4	R/W	1	T_SATA0_AHCI_CFG_ERR_CTRL_ENABLE_PXIS_IFS_D2H_SYNC_ESCAPE: 1h: ENABLE_PXIS_IFS_D2H_SYNC_ESCAPE_INIT (default)
3	R/W	1	T_SATA0_AHCI_CFG_ERR_CTRL_CLR_TX_BUFFER_ON_FATAL_ERROR: 1h: CLR_TX_BUFFER_ON_FATAL_ERROR_INIT (default)
2	R/W	1	T_SATA0_AHCI_CFG_ERR_CTRL_CLR_T2P_FIFO_ON_FATAL_ERROR: 1h: CLR_T2P_FIFO_ON_FATAL_ERROR_INIT (default)
1	R/W	1	T_SATA0_AHCI_CFG_ERR_CTRL_CLR_P2T_FIFO_ON_FATAL_ERROR: 1h: CLR_P2T_FIFO_ON_FATAL_ERROR_INIT (default)
0	R/W	1	T_SATA0_AHCI_CFG_ERR_CTRL_CLR_CMD_FIFO_ON_FATAL_ERROR: 1h: CLR_CMD_FIFO_ON_FATAL_ERROR_INIT (default)

T_SATA0_AHCI_HBA_CAP2_BKDR

This register has bit by bit backdoor fields for CAP2 register according to the AHCI 1.3 specification.

This register applies to SATA0 register space only.

Offset: 0x330 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:6	R	0h	T_SATA0_AHCI_HBA_CAP2_BKDR_RSVD_31_6: 0h: RSVD_31_6_ZERO
5	R/W	1	T_SATA0_AHCI_HBA_CAP2_BKDR_DEVSLP_ENTRY_FROM_SLUMBER_ONLY: NOTE: this field is reset by Cold Reset 1h: DEVSLP_ENTRY_FROM_SLUMBER_ONLY_TRUE (default) 0h: DEVSLP_ENTRY_FROM_SLUMBER_ONLY_FALSE
4	R/W	1	T_SATA0_AHCI_HBA_CAP2_BKDR_SUPPORTS_AGGR_DEVSLP: NOTE: this field is reset by Cold Reset 1h: SUPPORTS_AGGR_DEVSLP_TRUE (default) 0h: SUPPORTS_AGGR_DEVSLP_FALSE
3	R/W	1	T_SATA0_AHCI_HBA_CAP2_BKDR_SUPPORTS_DEVSLP: NOTE: this field is reset by Cold Reset 1h: SUPPORTS_DEVSLP_TRUE (default) 0h: SUPPORTS_DEVSLP_FALSE
2	R/W	1	T_SATA0_AHCI_HBA_CAP2_BKDR_AUTO_PARTIAL_TO_SLUMBER: NOTE: this field is reset by Cold Reset 1h: AUTO_PARTIAL_TO_SLUMBER_TRUE (default) 0h: AUTO_PARTIAL_TO_SLUMBER_FALSE

Bit	R/W	Reset	Description
1	R	0h	T_SATA0>_AHCI_HBA_CAP2_BKDR_RSVD_1: 0h: RSVD__ZERO
0	R/W	0	T_SATA0_AHCI_HBA_CAP2_BKDR_BOH: NOTE: this field is reset by Cold Reset 1h: BOH_TRUE 0h: BOH_FALSE (default)

T_SATA0_AHCI_HBA_CTL_1

This register applies to SATA0 register space only.

Offset: 0x338 | Read/Write: R/W | Reset: 0xC12994C0

Bit	R/W	Reset	Description
31	R/W	1	T_SATA0_AHCI_HBA_CTL_1_UNCLAMP_HBA_CTRLR: 1h: UNCLAMP_HBA_CTRLR_INIT (default)
30	R/W	1	T_SATA0_AHCI_HBA_CTL_1_INVALIDATE_PRDS_AT_FETCH: 1h: INVALIDATE_PRDS_AT_FETCH_INIT (default)
29	R/W	0	T_SATA0_AHCI_HBA_CTL_1_PREFETCH_TX_DATA_NO_DELAY: 0h: PREFETCH_TX_DATA_NO_DELAY_INIT (default)
28	R/W	0	T_SATA0_AHCI_HBA_CTL_1_CHK_L_IDLE_IN_WAKE_LINK: 0h: CHK_L_IDLE_IN_WAKE_LINK_INIT (default)
27	R/W	0	T_SATA0_AHCI_HBA_CTL_1_CHK_L_IDLE_IN_LOW_POWER: 0h: CHK_L_IDLE_IN_LOW_POWER_INIT (default)
26	R/W	0	T_SATA0_AHCI_HBA_CTL_1_SKIP_SCTL_DET_WR_COMRESET: NOTE: this field is reset by Cold Reset 0h: SKIP_SCTL_DET_WR_COMRESET_INIT (default)
25	R/W	0	T_SATA0_AHCI_HBA_CTL_1_ALWAYS_DENY_PMREQ: 0h: ALWAYS_DENY_PMREQ_INIT (default)
24	R/W	1	T_SATA0_AHCI_HBA_CTL_1_SKIP_SCTL_DET_WR_OFFLINE: NOTE: this field is reset by Cold Reset 1h: SKIP_SCTL_DET_WR_OFFLINE_INIT (default)
23:22	R/W	0	T_SATA0_AHCI_HBA_CTL_1_FETCH_REQ_TAKEN: 0h: FETCH_REQ_TAKEN_INIT (default)
21	R/W	1	T_SATA0_AHCI_HBA_CTL_1_NON_NCQ_IFS_TEMP_OFF: 1h: NON_NCQ_IFS_TEMP_OFF_INIT (default)
20	R/W	0	T_SATA0_AHCI_HBA_CTL_1_BM_HOLD_ACTV_TILL_INTR: 0h: BM_HOLD_ACTV_TILL_INTR_INIT (default)
19	R/W	1	T_SATA0_AHCI_HBA_CTL_1_PRD_SM_RXDATA_PREMATURE_END: 1h: PRD_SM_RXDATA_PREMATURE_END_INIT (default)
18	R/W	0	T_SATA0_AHCI_HBA_CTL_1_GOTO_PM_AGGR_FROM_P_IDLE: 1h: GOTO_PM_AGGR_FROM_P_IDLE_YES 0h: GOTO_PM_AGGR_FROM_P_IDLE_NO (default)
17	R/W	0	T_SATA0_AHCI_HBA_CTL_1_SEND_DATA_HDR_EARLY: 0h: SEND_DATA_HDR_EARLY_INIT (default)
16	R/W	1	T_SATA0_AHCI_HBA_CTL_1_FIS_SM_USE_RISEDGE_EOF: 1h: FIS_SM_USE_RISEDGE_EOF_INIT (default)
15	R/W	1	T_SATA0_AHCI_HBA_CTL_1_DETECT_B2B_NCQ_TAG_REPEAT: 1h: DETECT_B2B_NCQ_TAG_REPEAT_INIT (default)
14	R/W	0	T_SATA0_AHCI_HBA_CTL_1_NO_PRDBC_UPDATE: 0h: NO_PRDBC_UPDATE_INIT (default)

Bit	R/W	Reset	Description
13:12	R/W	1	T_SATA0_AHCI_HBA_CTL_1_ARB_PARK_MODE: 1h: ARB_PARK_MODE_LAST_GNTED (default) 0h: ARB_PARK_MODE_PORT0
11	R/W	0	T_SATA0_AHCI_HBA_CTL_1_ARB_GNT_CFG: 0h: ARB_GNT_CFG_INIT (default)
10	R/W	1	T_SATA0_AHCI_HBA_CTL_1_WAR_NDR_ACCEPT_ARC1_ISSUE: 1h: WAR_NDR_ACCEPT_ARC1_ISSUE_INIT (default)
9	R/W	0	T_SATA0_AHCI_HBA_CTL_1_DISABLE_BKGD_CMD_PREFETCH_2: 0h: DISABLE_BKGD_CMD_PREFETCH_2_INIT (default)
8	R/W	0	T_SATA0_AHCI_HBA_CTL_1_ISSUE_ORDER_CMD_SEL_IN_CBS: 0h: ISSUE_ORDER_CMD_SEL_IN_CBS_INIT (default)
7	R/W	1	T_SATA0_AHCI_HBA_CTL_1_INTR_PENDING_ON_CH_SEL: 1h: INTR_PENDING_ON_CH_SEL_INIT (default)
6	R/W	1	T_SATA0_AHCI_HBA_CTL_1_READ_OUT_BM_START: 1h: READ_OUT_BM_START_INIT (default)
5	R/W	0	T_SATA0_AHCI_HBA_CTL_1_SYNC_ESC_IN_ALPM: 0h: SYNC_ESC_IN_ALPM_INIT (default)
4	R/W	0	T_SATA0_AHCI_HBA_CTL_1_ENABLE_PRD_SPLIT_IN_CBS: 0h: ENABLE_PRD_SPLIT_IN_CBS_INIT (default)
3	R/W	0	T_SATA0_AHCI_HBA_CTL_1_DISABLE_BKGD_ACMD_PREFETCH: 0h: DISABLE_BKGD_ACMD_PREFETCH_INIT (default)
2	R/W	0	T_SATA0_AHCI_HBA_CTL_1_DISABLE_BKGD_DATA_PREFETCH: 0h: DISABLE_BKGD_DATA_PREFETCH_INIT (default)
1	R/W	0	T_SATA0_AHCI_HBA_CTL_1_DISABLE_BKGD_PRD_PREFETCH: 0h: DISABLE_BKGD_PRD_PREFETCH_INIT (default)
0	R/W	0	T_SATA0_AHCI_HBA_CTL_1_DISABLE_BKGD_CMD_PREFETCH_1: 0h: DISABLE_BKGD_CMD_PREFETCH_1_INIT (default)

T_SATA0_AHCI_HBA_PI_BKDR

This register applies to SATA0 register space only.

Offset: 0x33c | Read/Write: R/W | Reset: 0xXXXXXX01

Bit	R/W	Reset	Description
31:8	R	0h	T_SATA0_AHCI_HBA_PI_BKDR_31_8_RSVD: 0h: 31_8_RSVD_ZERO
7:0	R/W	1	T_SATA0_AHCI_HBA_PI_BKDR_PORTS_IMPL: NOTE: this field is reset by Cold Reset 0h: PORTS_IMPL_ZERO 3h: PORTS_IMPL_TWO Fh: PORTS_IMPL_FOUR 1h: PORTS_IMPL_DEFAULT (default)

T_SATA0_AHCI_HBA_PRE_STAGING_CONTROL

This register applies to SATA0 register space only.

Offset: 0x340 | Read/Write: R/W | Reset: 0xAA0006D4

Bit	R/W	Reset	Description
31:24	R/W	AAh	T_SATA0_AHCI_HBA_PRE_STAGING_CONTROL_SPARE: AAh: SPARE_INIT (default)

Bit	R/W	Reset	Description
23:20	R	0	T_SATA0_AHCI_HBA_PRE_STAGING_CONTROL_ENGAGED: 0h: ENGAGED_INIT (default)
19:18	R	0	Reserved
17:12	R/W	0	T_SATA0_AHCI_HBA_PRE_STAGING_CONTROL_DMA_XFER_CNT: 0h: DMA_XFER_CNT_INIT (default)
11:8	R/W	6h	T_SATA0_AHCI_HBA_PRE_STAGING_CONTROL_FIFO_LEVEL: 6h: FIFO_LEVEL_INIT (default)
7	R/W	1	T_SATA0_AHCI_HBA_PRE_STAGING_CONTROL_ENABLE: 1h: ENABLE_INIT (default)
6	R/W	1	T_SATA0_AHCI_HBA_PRE_STAGING_CONTROL_ATAPI_EN: 1h: ATAPI_EN_INIT (default)
5:0	R/W	14h	T_SATA0_AHCI_HBA_PRE_STAGING_CONTROL_PRDBC: 14h: PRDBC_INIT (default)

T_SATA<0>_CFG

Offset: 0x370 | Read/Write: R/W | Reset: 0x0F000000 (SATA) / 0x0F3XXXXX (SATA0)

Bit	R/W	Reset	Description
31:24	R/W	0Fh	T_SATA<0>_CFG_CTRL_TICKS_FOR_MASTER_TO: This is used in master slave emulation. Master waits for TICKS_FOR_MASTER_TO * 10**8 fpci_clk_sata cycles before timing out while waiting for response from slave drive on a device diagnostic command. Fh: CTRL_TICKS_FOR_MASTER_TO_250MHZ Fh: CTRL_TICKS_FOR_MASTER_TO_INIT (default)
23:0	0	R	Reserved (SATA only)
23	R/W	0	T_SATA0_CFG_WR_ALLOWED_SHAD_REG: This bit when set will allow writes to shadow register even when BSY/ DRQ is one 0h: WR_ALLOWED_SHAD_REG_INIT (default)
22	R/W	0	T_SATA0_CFG_HOT_RESET_ON_BM_START_ALONE: 0h: HOT_RESET_ON_BM_START_ALONE_INIT (default)
21	R/W	1	T_SATA0_CFG_HOT_RESET_OLD_BEHAVIOUR: 1h: HOT_RESET_OLD_BEHAVIOUR_INIT (default)
20	R/W	1	T_SATA0_CFG_RESET_SREGS_ON_OOB_COMRESET: NOTE: this field is reset by Cold Reset 1h: RESET_SREGS_ON_OOB_COMRESET_TRUE (default) 0h: RESET_SREGS_ON_OOB_COMRESET_FALSE
19:0	R	0h	T_SATA0_CFG_RSVD_1: 0h: RSVD_1_VAL

T_SATA0_CTL_SHADOW

This register applies to SATA0 register space only.

Offset: 0x378 | Read/Write: R/W | Reset: 0x3XXXXXXX

Bit	R/W	Reset	Description
31	R/W	0	T_SATA0_CTL_SHADOW_PIO_DATA_READ_RETRY: 0h: PIO_DATA_READ_RETRY_NEW (default) 1h: PIO_DATA_READ_RETRY_OLD
30	R/W	0	T_SATA0_CTL_SHADOW_ALLOW_CTRL_WR_WHEN_DRIVE_NOT_PRESENT: 1h: ALLOW_CTRL_WR_WHEN_DRIVE_NOT_PRESENT_YES 0h: ALLOW_CTRL_WR_WHEN_DRIVE_NOT_PRESENT_NO (default)

Bit	R/W	Reset	Description
29	R/W	1	T_SATA0_CTL_SHADOW_RET_ERROR_7F_WHEN_MASTER_ABSENT: 1h: RET_ERROR_7F_WHEN_MASTER_ABSENT_TRUE (default) 0h: RET_ERROR_7F_WHEN_MASTER_ABSENT_NO
28	R/W	1	T_SATA0_CTL_SHADOW_FIX_SRST_WHEN_ONE_DEVICE_NOT_PRESENT: 1h: FIX_SRST_WHEN_ONE_DEVICE_NOT_PRESENT_INIT (default)
27	R/W	0	T_SATA0_CTL_SHADOW_USE_ELONGATED_FIFO_HOT_RESET: 0h: USE_ELONGATED_FIFO_HOT_RESET_INIT (default)
26:0	R	0h	T_SATA0_CTL_SHADOW_RSVD: 0h: RSVD_VAL

T_SATA0_CFG_FPCI_0

This register applies to SATA0 register space only.

Offset: 0x430 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31	R/W	0	T_SATA0_CFG_FPCI_0_AHCI_MODE_DEP_ON_EN: When class code is 0106, ahci_mode will be one When class code is 0101, ahci_mode will be zero in other cases ahci_mode will depend on ahci_en if this bit is set, if bit is zero ahci_mode will be one NOTE: this field is reset by Cold Reset 1h: AHCI_MODE_DEP_ON_EN_YES 0h: AHCI_MODE_DEP_ON_EN_NO (default)
30	R/W	1	T_SATA0_CFG_FPCI_0_SEND_MSG_ON_NEW_INTR: This is used in the case of multiple MSI. This is to disbale/ enable sending interrupts when a new bit is written in pxis and if a some bits are cleared but not all and there is a posedge on a bit in pxie (see the AHCI specification, section 10.7.2) 1h: SEND_MSG_ON_NEW_INTR_INIT (default)
29	R/W	1	T_SATA0_CFG_FPCI_0_SEND_NEW_MSG_IS_NOT_ZERO: This is used in the case of a single MSI to send new message (intr), if software clears some bits of IS but not all (see the AHCI specification, section 10.7.2) 1h: SEND_NEW_MSG_IS_NOT_ZERO_INIT (default)
28:0	R	0h	T_SATA0_CFG_FPCI_0_RSVD: 0h: RSVD_VAL

T_SATA<0>_IDE1

When 40-bit addressing mode is enabled, bits [39:32] of the prd_base are loaded.

Offset: 0x490 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	T_SATA<0>_IDE1_OLD_IDE_TIMING: 0h: OLD_IDE_TIMING_DEFAULT (default)

T_SATA0_CFG_ESATA_CTRL

This register implements settings for controlling behavior of the AHCI SM during error conditions.

This register applies to SATA0 register space only.

Offset: 0x494 | Read/Write: R/W | Reset: 0xXXXX0000

Bit	R/W	Reset	Description
31:16	R	0h	T_SATA0_CFG_ESATA_CTRL_RSVD:

Bit	R/W	Reset	Description
			0h: RSVD_VAL
15:8	R/W	0	T_SATA0_CFG_ESATA_CTRL_TX_PEAK_LIMIT: NOTE: this field is reset by Cold Reset 0h: TX_PEAK_LIMIT_INIT (default)
7:0	R/W	0	T_SATA0_CFG_ESATA_CTRL_TX_AMP_LIMIT: NOTE: this field is reset by Cold Reset 0h: TX_AMP_LIMIT_INIT (default)

T_SATA<0>_FEATURE

Feature Register

The feature options are shown in the following table.

Bit	Feature
31	NBP enable
30:13	Reserved
12:11	RAID function by SW programming
10	Reserved
9	Only 4 SATA ports supported
8	Software feature
7	Only 2 SATA ports supported
6	AHCI disabled
5	eSATA disabled
4	Aggressive power management disabled
3	Staggered spin-up disabled
2	iSCSI disabled
1	SATA Gen2 enabled
0	Advanced storage features disabled

Offset: 0x498 | Read/Write: R/W | Reset: 0xXX00XXXX

Bit	R/W	Reset	Description
31:24	R	0h	T_SATA<0>_FEATURE_RSVD_31_24: 0h: RSVD_31_24_VAL
23:18	R	0	T_SATA<0>_FEATURE_AHCI_ESATA_DIS: 3Fh: AHCI_ESATA_DIS_YES 0h: AHCI_ESATA_DIS_NO (default)
17:14	R	0	Reserved
13	R	0h	T_SATA<0>_FEATURE_RSVD_13: 0h: RSVD_13_VAL
12:10	R	None	T_SATA<0>_FEATURE_RAID_FUNCTION_DIS:
9	R	0	T_SATA<0>_FEATURE_PORTS_2_3_DIS: 1h: PORTS_2_3_DIS_YES 0h: PORTS_2_3_DIS_NO (default)
8	R	0	T_SATA<0>_FEATURE_PORTS_0_1_DIS: 1h: PORTS_0_1_DIS_YES 0h: PORTS_0_1_DIS_NO (default)

Bit	R/W	Reset	Description
7	R	0	Reserved
6	R	0h	T_SATA<0>_FEATURE_RSVD_6: 0h: RSVD_6_VAL 0h: RSVD_7_VAL
5	R	0h	T_SATA<0>_FEATURE_RSVD_5: 0h: RSVD_5_VAL
4	R	0	T_SATA<0>_FEATURE_AHCI_POWER_DIS: 1h: AHCI_POWER_DIS_YES 0h: AHCI_POWER_DIS_NO (default)
3	R	0	T_SATA<0>_FEATURE_FBS_DIS: 1h: FBS_DIS_YES 0h: FBS_DIS_NO (default)
2	R	0	T_SATA<0>_FEATURE_GEN3_EN: 1h: GEN3_EN_YES 0h: GEN3_EN_NO (default)
1	R	1	T_SATA<0>_FEATURE_GEN2_EN: 1h: GEN2_EN_YES (default) 0h: GEN2_EN_NO
0	R	0h	T_SATA<0>_FEATURE_RSVD_0: 0h: RSVD_0_VAL

T_SATA<0>_CTL1

Miscellaneous CTL1

Offset: 0x4a0 | Read/Write: R/W | Reset: 0xXXXXXX01

Bit	R/W	Reset	Description
31	R	0	Reserved
30:10	R	0h	T_SATA<0>_CTL1_RSVD_30_10: 0h: RSVD_30_10_VAL
9	R/W	1	T_SATA<0>_CTL1_BLK_NONPIO_IDP: BLK_NONPIO_IDP is used to suppress the nonpio read in the case of idp read. At default 1 value the second read will be blocked. At 0 both reads will occur. 1h: BLK_NONPIO_IDP_DEFAULT (default)
8:1	R	0	Reserved
0	R/W	1	T_SATA<0>_CTL1_SATA_ADNVCD_SPD_NEGO: ADNVCD_SPD_NEGO turns on an advanced way of detecting sata speeds. SATA will mix detecting the two supported speeds. Otherwise it detects gen2 first and then gen1. NOTE: this field is reset by Cold Reset 1h: SATA_ADNVCD_SPD_NEGO_YES (default) 0h: SATA_ADNVCD_SPD_NEGO_NO

T_SATA<0>_BKDOOR_CC

This register is used to backdoor update the programming interface field and class code of the register at address 0x8 (T_SATA<0>_CFG_2[15:8]). Note that the default value of the programming interface field is 0x85 for SATA<0> and 0x8A for SATA1.

To update the programming interface field, the BIOS needs to do the following:

1. Read T_SATA<0>_CFG_SATA_BACKDOOR_PROG_IF_EN - Address 54C

2. Write 1 to T_SATA<0>_CFG_SATA_BACKDOOR_PROG_IF_EN

3. Write the desired programming interface value to T_SATA<0>_BKDOOR_CC[15:8]

4. Restore T_SATA<0>_CFG_SATA_BACKDOOR_PROG_IF_EN to the value read in step#1

Offset: 0x4a4 | Read/Write: R/W | Reset: 0x01018AXX (SATA) / 0x010185XX (SATA0)

Bit	R/W	Reset	Description
31:16	R/W	0101h	T_SATA<0>_BKDOOR_CC_CLASS_CODE: NOTE: this field is reset by Cold Reset 101h: CLASS_CODE_INIT (default)
15:8	R/W	8A (SATA) 85h (SATA0)	T_SATA<0>_BKDOOR_CC_PROG_IF: NOTE: this field is reset by Cold Reset 85h: PROG_IF_INIT (default)
7:0	R	0h	T_SATA<0>_BKDOOR_CC_RSVD_0: 0h: RSVD_0_VAL

T_SATA<0>_CFG_CTRL_1

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

Bit	R/W	Reset	Description
31:2	R	0	Reserved
1	R/W	0	T_SATA<0>_CFG_CTRL_1_SATA_CAP: This bit controls whether or not SATA capability is enabled or not. By default it is enabled. NOTE: this field is reset by Cold Reset 1h: SATA_CAP_DISABLE 0h: SATA_CAP_ENABLE (default)
0	R/W	0	T_SATA<0>_CFG_CTRL_1_48BIT_ADDRESS_DISABLE: 48BIT_ADDRESS_DISABLE is used to disable 48-bit addressing mode. The SATA implementation of 48-bit addressing is such that this bit is no longer needed. It can be removed safely and is currently ignored in SATA. NOTE: this field is reset by Cold Reset 0h: 48BIT_ADDRESS_DISABLE_NO (default) 1h: 48BIT_ADDRESS_DISABLE_YES

T_SATA0_CFG_POWER_GATE

This register applies to SATA0 register space only.

Offset: 0x4ac | Read/Write: R/W | Reset: 0xXX000000

Bit	R/W	Reset	Description
31:24	R	0h	T_SATA0_CFG_POWER_GATE_RSVD: 0h: RSVD_VAL
23	R/W	0	T_SATA0_CFG_POWER_GATE_SSTS_RESTORED: 0h: SSTS_RESTORED_DEFAULT (default) 1h: SSTS_RESTORED_YES
22	R	0	T_SATA0_CFG_POWER_GATE_PHY_NOT_IN_SLUMBER: 0h: PHY_NOT_IN_SLUMBER_INIT (default)
21	R/W	0	T_SATA0_CFG_POWER_GATE_STOP_INTERRUPTS: 0h: STOP_INTERRUPTS_INIT (default)
20	R	0	T_SATA0_CFG_POWER_GATE_COUNT_FOR_PADS_DONE: 0h: COUNT_FOR_PADS_DONE_INIT (default)
19	R/W	0	T_SATA0_CFG_POWER_GATE_PADS_POWER_DOWN: 0h: PADS_POWER_DOWN_INIT (default)
18	R/W	0	T_SATA0_CFG_POWER_GATE_HW_WAKEUP_SUPPORT: 1h: HW_WAKEUP_SUPPORT_YES

Bit	R/W	Reset	Description
			0h: HW_WAKEUP_SUPPORT_NO (default)
17:2	R	0	T_SATA0_CFG_POWER_GATE_SM2SATA_PG_INFO: Indicates the data stored in SM unit - for now it has nothing 0h: SM2SATA_PG_INFO_DEFAULT (default)
1	R/W	0	T_SATA0_CFG_POWER_GATE_POWER_UNGATE_COMP: SW writes this bit to one when SW has restored all the registers necessary HW will write it back to zero 1h: POWER_UNGATE_COMP_YES 0h: POWER_UNGATE_COMP_NO (default) 1h: POWER_UNGATE_COMP_SET
0	R/W	0	T_SATA0_CFG_POWER_GATE_ENTER_PG: SW writes this bit to one when it wants the controller to enter Power Gating HW will write it back to zero 1h: ENTER_PG_YES 0h: ENTER_PG_NO (default) 1h: ENTER_PG_SET

T_SATA0_CFG_CTL_GLUE

This register applies to SATA0 register space only.

Offset: 0x4b0 | Read/Write: R/W | Reset: 0xXXXXXXCB

Bit	R/W	Reset	Description
31:8	R	0h	T_SATA0_CFG_CTL_GLUE_RSVD: 0h: RSVD_VAL
7	R/W	1	T_SATA0_CFG_CTL_GLUE_DEVSLP_WITH_GHC_HR_PXCMD_ST: NOTE: this field is reset by Cold Reset 1h: DEVSLP_WITH_GHC_HR_PXCMD_ST_INIT (default)
6	R/W	1	T_SATA0_CFG_CTL_GLUE_RESET_COMMA_DETECTION_CNTR: NOTE: this field is reset by Cold Reset 1h: RESET_COMMA_DETECTION_CNTR_INIT (default)
5	R/W	0	T_SATA0_CFG_CTL_GLUE_HOLD_REQ_DEVCLK_CLAMP_2: NOTE: this field is reset by Cold Reset 0h: HOLD_REQ_DEVCLK_CLAMP_2_INIT (default)
4	R/W	0	T_SATA0_CFG_CTL_GLUE_HOLD_REQ_DEVCLK_CLAMP_1: NOTE: this field is reset by Cold Reset 0h: HOLD_REQ_DEVCLK_CLAMP_1_INIT (default)
3	R/W	1	T_SATA0_CFG_CTL_GLUE_LOCKDET_OVR_LOGIC_FIX: NOTE: this field is reset by Cold Reset 1h: LOCKDET_OVR_LOGIC_FIX_INIT (default)
2	R/W	0	T_SATA0_CFG_CTL_GLUE_OVERRIDE_LOCKDET_FOR_NVA: NOTE: this field is reset by Cold Reset 0h: OVERRIDE_LOCKDET_FOR_NVA_INIT (default)
1	R/W	1	T_SATA0_CFG_CTL_GLUE_USE_OVERRIDE_LOCKDET: NOTE: this field is reset by Cold Reset 1h: USE_OVERRIDE_LOCKDET_YES (default) 0h: USE_OVERRIDE_LOCKDET_NO
0	R/W	1	T_SATA0_CFG_CTL_GLUE_FIX_FPCI_WRDAT: When set to one adds a clock delay which is necessary for correct operation This bit will be removed once the platform emulation could do writes to BAR5 space NOTE: this field is reset by Cold Reset 1h: FIX_FPCI_WRDAT_YES (default) 0h: FIX_FPCI_WRDAT_NO

T_SATA0_CFG_CTL_FA

This register applies to SATA0 register space only.

Offset: 0x4bc | Read/Write: R/W | Reset: 0x00000083

Bit	R/W	Reset	Description
31:8	R	0	Reserved
7:3	R/W	10h	T_SATA0_CFG_CTL_FA_CFG_EATER: 10h: CFG_EATER_DEFAULT (default)
2	R/W	0	T_SATA0_CFG_CTL_FA_USE_CFG_EATER: 1h: USE_CFG_EATER_YES 0h: USE_CFG_EATER_NO (default)
1	R/W	1	T_SATA0_CFG_CTL_FA_EAT_CLK: 1h: EAT_CLK_YES (default) 0h: EAT_CLK_NO
0	R/W	1	T_SATA0_CFG_CTL_FA_FPCI_CLK_IS_128MHZ: 1h: FPCI_CLK_IS_128MHZ_YES (default) 0h: FPCI_CLK_IS_128MHZ_NO

T_SATA<0>_PERF0

Offset: 0x4f0 | Read/Write: R/W | Reset: 0x00000800

Bit	R/W	Reset	Description
31:29	R	0	Reserved
28	R/W	0	T_SATA<0>_PERF0_DONT_DELAY_ROK: Setting DONT_DELAY_ROK turns on an optimization feature that should help performance on DMA reads. Transport layer then completes the current transfer even when busmaster has not yet drained all of the Receive FIFO. Setting this bit to _TRUE should result in better performance and therefore it is the preferred mode of operation. This is not the default Reset value and it is expected that the BIOS or Driver will enable this feature. NOTE: this field is reset by Cold Reset 0h DONT_DELAY_ROK_FALSE (default) 1h: DONT_DELAY_ROK_TRUE
27	R	0	Reserved
26:25	R/W	0	T_SATA<0>_PERF0_SATA_PLL_POWERDOWN: NOTE: this field is reset by Cold Reset 1h: SATA_PLL_POWERDOWN_YES 0h SATA_PLL_POWERDOWN_NO (default)
24:12	R	0	Reserved
11:7	R/W	10h	T_SATA<0>_PERF0_MIN_FPCI_CLK_FREQ: This field set the minimum frequency at which the fpci_clk should run in increments of 1/16th. It provides SW to override value set by frequency adjuster unit. Note that the actual frequency level is maximum of all the controllers. NOTE: this field is reset by Cold Reset 10h MIN_FPCI_CLK_FREQ_16 (default)
6:0	R	0	Reserved

T_SATA<0>_PERF1

This register is used to configure some of the SATA implementation options.

Offset: 0x4f4 | Read/Write: R/W | Reset: 0x00005A6E

Bit	R/W	Reset	Description
31:24	R/W	0	T_SATA<0>_PERF1_MAX_WATER_MARK:

Bit	R/W	Reset	Description
			The MAX_WATER_MARK field tracks the maximum occupancy of the two i2f_fifos. The purpose of this field is to give us guidance on the setting of HIGH_WATER_MARK field. Ideally, the HIGH_WATER_MARK field should be set so that the MAX_WATER_MARK field almost hit full. Note that MAX_WATER_MARK field is only 8 bits wide, so it can only track the lower 8 bits of the fifo level. The MSB value can be inferred from the MSB value of the 9 bit HIGH_WATER_MARK field. T_SATA<0>_CFG_39_RXDATA_IB_DELAY[0] enables resetting SSTATUS_DET and SSTATUS_SPD fields to 0 when channels enter into partial or slumber low power modes. By default this bit is 0 and these fields are not reset. NOTE: this field is reset by Cold Reset 0h MAX_WATER_MARK_0 (default)
23:16	R	0	Reserved
15:8	R/W	5Ah	T_SATA<0>_PERF1_HIGH_WATER_MARK_PIO: Bit 13:8 of the register is used for PIO reads high water mark. For PIO reads, fifo depth is 48. But each entry is 32bit, that is it contains only one dword. So, if mark is set at Y, there is headroom for (48-Y) dwords for drive to react to us sending HOLD. 5Ah HIGH_WATER_MARK_PIO_INIT (default)
7:0	R/W	6Eh	T_SATA<0>_PERF1_HIGH_WATER_MARK: This field configures the read fifo limit at which point the host starts sending HOLD primitives to back-off the data transfer from device. This field is only relevant during READs and should typically be set to as high a value without running the risk of overflowing the Fifo. Bit5:0 of the register is used for DMA read high water mark. For DMA reads, fifo depth is 48. And, each entry is 64bit, that is it contains two dwords. So, if the watermark is set at X then there is headroom for (48-X)*2 dwords for drive to react to controller sending HOLD. 6Eh HIGH_WATER_MARK_INIT (default)

T_SATA0_SPARE_0

Spare register on cold reset with zero as the default value.

This register applies to SATA0 register space only.

Offset: 0x520 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	T_SATA0_SPARE_0_RSVD: 0h RSVD_DEFAULT (default)

T_SATA0_SPARE_1

Spare register on warm reset with zero as the default value.

This register applies to SATA0 register space only.

Offset: 0x524 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	T_SATA0_SPARE_1_RSVD: 0h RSVD_DEFAULT (default)

T_SATA0_SPARE_2

Spare register on cold reset with 1 as the default value.

This register applies to SATA0 register space only.

Offset: 0x528 | Read/Write: R/W | Reset: 0xFFFFFFFF

Bit	R/W	Reset	Description

Bit	R/W	Reset	Description
31:7	R/W	1FFFFFFh	T_SATA0_SPARE_2_RSVD: 1FFFFFFh RSVD_DEFAULT (default)
6	R/W	1	T_SATA0_SPARE_2_SPLIT_D2H_REG_FIS: 1h SPLIT_D2H_REG_FIS_DEFAULT (default)
5	R/W	1	T_SATA0_SPARE_2_LATCH_PXCMD_FRE: 1h LATCH_PXCMD_FRE_DEFAULT (default)
4	R/W	1	T_SATA0_SPARE_2_SET_PXCMD_FR_ON_FRE: 1h SET_PXCMD_FR_ON_FRE_DEFAULT (default)
3	R/W	1	T_SATA0_SPARE_2_ERR_CLEANUP_WAIT_P_IDLE: 1h ERR_CLEANUP_WAIT_P_IDLE_DEFAULT (default)
2	R/W	1	T_SATA0_SPARE_2_FIS_PRO_WAIT_P_IDLE: 1h FIS_PRO_WAIT_P_IDLE_DEFAULT (default)
1	R/W	1	T_SATA0_SPARE_2_WAIT_DATA_DFIFO_POP: 1h WAIT_DATA_DFIFO_POP_DEFAULT (default)
0	R/W	1	T_SATA0_SPARE_2_REM_TWO_ALIGNS_IN_BIST_L: 1h REM_TWO_ALIGNS_IN_BIST_L_DEFAULT (default)

T_SATA0_SPARE_3

Spare register on warm reset with 1 as the default value.

This register applies to SATA0 register space only.

Offset: 0x52c | Read/Write: R/W | Reset: 0x0001A5E0

Bit	R/W	Reset	Description
31:19	R/W	0	T_SATA0_SPARE_3_RSVD: 0h RSVD_DEFAULT (default)
18:0	R/W	1A5E0h	T_SATA0_SPARE_3_COUNT_FOR_1_MS: 1A5E0h COUNT_FOR_1_MS_DEFAULT (default)

T_SATA<0>_CHXCFG1

Offset: 0x530 | Read/Write: R/W | Reset: 0xXXXX00XX

Bit	R/W	Reset	Description
31:16	R	0h	T_SATA<0>_CHXCFG1_RSVD_1: 0h RSVD_1_VAL
15:12	R/W	0	T_SATA<0>_CHXCFG1_PHY_CHX_RX_CTL: NOTE: this field is reset by Cold Reset 0h PHY_CHX_RX_CTL_INIT (default) 0h PHY_CHX_RX_CTL_VAL_0 1h PHY_CHX_RX_CTL_VAL_1 2h PHY_CHX_RX_CTL_VAL_2 3h PHY_CHX_RX_CTL_VAL_3 Fh PHY_CHX_RX_CTL_VAL_F
11:8	R/W	0	T_SATA<0>_CHXCFG1_PHY_CHX_TX_CTL: NOTE: this field is reset by Cold Reset 0h PHY_CHX_TX_CTL_INIT (default) 0h PHY_CHX_TX_CTL_VAL_0 1h PHY_CHX_TX_CTL_VAL_1 2h PHY_CHX_TX_CTL_VAL_2 3h PHY_CHX_TX_CTL_VAL_3 Fh PHY_CHX_TX_CTL_VAL_F
7:2	R	0h	T_SATA<0>_CHXCFG1_RSVD_0:

Bit	R/W	Reset	Description
			0h RSVD_0_VAL
1	R/W	0	T_SATA<0>_CHXCFG1_CHX_NEAR_LOOP_BACK: NOTE: this field is reset by Cold Reset 0h CHX_NEAR_LOOP_BACK_INACTIVE 1h CHX_NEAR_LOOP_BACK_ACTIVE 0h CHX_NEAR_LOOP_BACK_DEFAULT (default)
0	R/W	0	T_SATA<0>_CHXCFG1_CHX_RESET: This can be used to reset the port in IDE mode of SATA controller. It is not advisable to use it in AHCI mode. NOTE: this field is reset by Cold Reset 0h CHX_RESET_INACTIVE 1h CHX_RESET_ACTIVE 0h CHX_RESET_DEFAULT (default)

T_SATA<0>_PHY_CTRL

PLL_SLEEP describes power setting of SATA2 internal PHY. Input to PHY is actually most active power setting from two SATA controllers. TX/RX power states are per port.

Offset: 0x534 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:5	R	0h	T_SATA<0>_PHY_CTRL_RSVD_1: 0h RSVD_1_VAL
4:3	R/W	0	T_SATA<0>_PHY_CTRL_PLL_SLEEP: NOTE: this field is reset by Cold Reset 0h PLL_SLEEP_INIT (default) 0h PLL_SLEEP_ACTIVE 1h PLL_SLEEP_3G_DISABLED 2h PLL_SLEEP_ALL_CLKS_DISABLED 3h PLL_SLEEP_PLL_DISABLED
2:1	R	0h	T_SATA<0>_PHY_CTRL_RSVD_0: 0h RSVD_0_VAL
0	R/W	1	T_SATA<0>_PHY_CTRL_SLUMBER_DURING_D3: NOTE: this field is reset by Cold Reset 1h SLUMBER_DURING_D3_ENABLE (default) 0h SLUMBER_DURING_D3_DISABLE

T_SATA<0>_LDT

LDT Unit ID Register

The CFG_19 register contains the LDT Unit ID(s) for the Device. The Unit IDs are used by the device as a transaction ID on the LDT interface to uniquely identify where the request originated. Some devices will support 2 Unit IDs, 1 for isochronous requests and the other for non-isochronous requests.

Offset: 0x53c | Read/Write: R/W | Reset: 0x000000XX

Bit	R/W	Reset	Description
31:13	R	0	Reserved
12:8	R/W	0	T_SATA<0>_LDT_NON_ISO_UNIT_ID: Non-ISO Unit ID. 0h NON_ISO_UNIT_ID_0 (default) 9h NON_ISO_UNIT_ID_9 1h NON_ISO_UNIT_ID_1 2h NON_ISO_UNIT_ID_2 1Fh NON_ISO_UNIT_ID_31

Bit	R/W	Reset	Description
7:5	R	0	Reserved
4:0	R	0h	T_SATA<0>_LDT_ISO_UNIT_ID: ISO Unit ID. Hard-wired to 0. 0h ISO_UNIT_ID_0 1h ISO_UNIT_ID_1 2h ISO_UNIT_ID_2 1Fh ISO_UNIT_ID_31

T_SATA<0>_CTRL

Offset: 0x540 | Read/Write: R/W | Reset: 0x60300007

Bit	R/W	Reset	Description
31	R	0	Reserved
30	R/W	1	T_SATA<0>_CTRL_ENABLE_CH2_CH4_INTR: NOTE: this field is reset by Cold Reset 1h ENABLE_CH2_CH4_INTR_YES (default) 0h ENABLE_CH2_CH4_INTR_NO
29	R/W	1	T_SATA<0>_CTRL_ENABLE_CH1_CH3_INTR: NOTE: this field is reset by Cold Reset 1h ENABLE_CH1_CH3_INTR_YES (default) 0h ENABLE_CH1_CH3_INTR_NO
28	R	0	Reserved
27	R/W	0	T_SATA<0>_CTRL_NP_RSP_ERROR_INTR: NOTE: this field is reset by Cold Reset 1h NP_RSP_ERROR_INTR_YES 0h NP_RSP_ERROR_INTR_NO (default)
26	R/W	0	T_SATA<0>_CTRL_PW_CMD_ERROR_INTR: NOTE: this field is reset by Cold Reset 1h PW_CMD_ERROR_INTR_YES 0h PW_CMD_ERROR_INTR_NO (default)
25	R/W	0	T_SATA<0>_CTRL_NP_RSP_ERROR_INTR_EN: NOTE: this field is reset by Cold Reset 1h NP_RSP_ERROR_INTR_EN_YES 0h NP_RSP_ERROR_INTR_EN_NO (default)
24	R/W	0	T_SATA<0>_CTRL_PW_CMD_ERROR_INTR_EN: NOTE: this field is reset by Cold Reset 1h PW_CMD_ERROR_INTR_EN_YES 0h PW_CMD_ERROR_INTR_EN_NO (default)
23:22	R	0	Reserved
21	R/W	1	T_SATA<0>_CTRL_CH4_EN: NOTE: this field is reset by Cold Reset 1h CH4_EN_YES (default) 0h CH4_EN_NO
20	R/W	1	T_SATA<0>_CTRL_CH3_EN: NOTE: this field is reset by Cold Reset 1h CH3_EN_YES (default) 0h CH3_EN_NO
19:4	R	0	Reserved
3	R/W	0	T_SATA<0>_CTRL_INTF_ERR_HANDLING_EN: This is used to enable an advanced feature to handle SATA interface errors. It includes detecting premature CRC in non-AHCI mode and resetting transport layer on comreset. The interface errors are mostly seen on emulation system and are not expected on silicon. NOTE: this field is reset by Cold Reset 1h INTF_ERR_HANDLING_EN_YES

Bit	R/W	Reset	Description
			0h INTF_ERR_HANDLING_EN_NO (default)
2	R/W	1	T_SATA<0>_CTRL_BAR5_SPACE_EN: BAR5_SPACE_EN bit enables use of BAR5 space. So, if this bit is written with zero then SATA will not accept any cycles to BAR5 (starting from 1KB to 4KB). By default this bit is low. NOTE: this field is reset by Cold Reset 0h BAR5_SPACE_EN_NO 1h BAR5_SPACE_EN_YES (default)
1	R/W	1	T_SATA<0>_CTRL_PRI_CHANNEL_EN: Enables the primary SATA channel. Value 1 denotes that the respective channel is enabled and 0 indicates disabled channel. NOTE: this field is reset by Cold Reset 0h PRI_CHANNEL_EN_NO 1h PRI_CHANNEL_EN_YES (default)
0	R/W	1	T_SATA<0>_CTRL_SEC_CHANNEL_EN: Enables the secondary SATA channel. Value 1 denotes that the respective channel is enabled and 0 indicates disabled channel. NOTE: this field is reset by Cold Reset 0h SEC_CHANNEL_EN_NO 1h SEC_CHANNEL_EN_YES (default)

T_SATA<0>_CFG_SATA

Offset: 0x54c | Read/Write: R/W | Reset: 0xXXXXX02X

Bit	R/W	Reset	Description
31:17	R	0h	T_SATA<0>_CFG_SATA_RSVD_1: 0h RSVD_1_VAL
16	R/W	0 (SATA) 1 (SATA0)	T_SATA<0>_CFG_SATA_FORCE_NATIVE: NOTE: this field is reset by Cold Reset 1h FORCE_NATIVE_ENABLED (SATA0 default) 0h FORCE_NATIVE_DISABLED (SATA default)
15	R/W	0	T_SATA<0>_CFG_SATA_CTRL_ALT_UID_SCHEME: NOTE: this field is reset by Cold Reset 0h CTRL_ALT_UID_SCHEME_INIT (default)
14	R	Unknown	T_SATA<0>_CFG_SATA_CTRL_SATA_GEN3_CAPABLE: 1h CTRL_SATA_GEN3_CAPABLE_YES 0h CTRL_SATA_GEN3_CAPABLE_NO
13	R	Unknown	T_SATA<0>_CFG_SATA_CTRL_SATA_GEN2_CAPABLE: SATA_GEN2_CAPABLE is read-only register that reflects the value of bond_sata2. If high, GEN2 speed is enabled, if low, only GEN1 speed is available. This register is also used to send near end loopback and reset the channels. The MIN_ON_TIME bits set the additional time the activity led remains lit after a transaction. This keeps the LED from being a dim flicker during short burst. 1h CTRL_SATA_GEN2_CAPABLE_YES 0h CTRL_SATA_GEN2_CAPABLE_NO
12	R/W	0	T_SATA<0>_CFG_SATA_BACKDOOR_PROG_IF_EN: NOTE: this field is reset by Cold Reset 1h BACKDOOR_PROG_IF_EN_YES 0h BACKDOOR_PROG_IF_EN_NO (default)
11:7	R/W	0	T_SATA<0>_CFG_SATA_PORT2UNITID_MAPPING: NOTE: this field is reset by Cold Reset 0h PORT2UNITID_MAPPING_DEFAULT (default)
6	R/W	0	T_SATA<0>_CFG_SATA_MSI_CAP_DISABLE: Normally, the T_SATA<0>_CFG_17_NEXT_PTR field points to T_SATA<0>_MSI_CTRL (0xB0). However, if SATA_MSI_CAP_DISABLE is set to _YES, then the T_SATA<0>_CFG_17_NEXT_PTR field would point to _NULL. If future capabilities are added beyond the MSI pointer, then care should be taken to ensure that CFG_17_NEXT_PTR bypasses the T_SATA<0>_MSI_CTRL address and points to the next capabilities pointer instead of _NULL.

Bit	R/W	Reset	Description
			NOTE: this field is reset by Cold Reset 0h MSI_CAP_DISABLE_NO (default) 1h MSI_CAP_DISABLE_YES
5	R/W	1	T_SATA<0>_CFG_SATA_MSIX_CAP_DISABLE: 0h MSIX_CAP_DISABLE_NO 1h MSIX_CAP_DISABLE_YES (default)
4	R/W	0	T_SATA<0>_CFG_SATA_USE_40B_ADDR: If this bit is set then SATA uses NVIDIA style 40b addressing while DMAing data. It gets upper 8 bits from the 23:16 bits of the prd count associated with the prd address. NOTE: this field is reset by Cold Reset 0h USE_40B_ADDR_NO (default) 1h USE_40B_ADDR_YES
3	R/W	0	T_SATA<0>_CFG_SATA_ERROR_HANDLING: NOTE: this field is reset by Cold Reset 0h ERROR_HANDLING_NO (default) 1h ERROR_HANDLING_YES
2	R/W	0	T_SATA<0>_CFG_SATA_CTRL_RAID_MODE_CTRL: CTRL_RAID_MODE was originally intended to allow customers to downgrade their RAID capability in the SBIOS. Setting this bit to 0 would allow only RAID 0 and RAID 1 support in the driver while setting to 1 would allow all RAID support. Setting this bit to 1 modifies the DeviceID of the SATA controller which would be a key to the driver. NOTE: this field is reset by Cold Reset 0h CTRL_RAID_MODE_CTRL_OEM (default) 1h CTRL_RAID_MODE_CTRL_CHANNEL
1	R	Unknown	T_SATA<0>_CFG_SATA_CTRL_STORAGE_FEATURE: STORAGE_FEATURE goes to both SATA and IDE config spaces. This is a general purpose bit that SW can use to distinguish between Advanced and Basic storage controller modes. This bit will be interpreted to either allow or not allow RAID 0+1 or 5 mode. ADVANCED mode will allow all RAID modes and BASIC will allow only RAID 0 and RAID 1 modes. It is strongly encouraged to keep the meaning of this STORAGE_FEATURE bit the same for future chips and to create new bits if necessary. This will allow SW compatibility. 1h CTRL_STORAGE_FEATURE_BASIC 0h CTRL_STORAGE_FEATURE_ADVANCED
0	R	0h	T_SATA<0>_CFG_SATA_RSVD_0: 0h RSVD_0_VAL

T_SATA<0>_CFG_MISC

Offset: 0x550 | Read/Write: R/W | Reset: 0x00000163

Bit	R/W	Reset	Description
31:12	R	0	Reserved
11:8	R/W	1	T_SATA<0>_CFG_MISC_PHY_RESET_USAGE_MODE: Bit 0 controls use of lockdet as phy reset. Bit 1 when high lets us update ata status reg quickly during pio-writes. By default its high. Bit 2 controls pi_reset, it controls whether to use lockdet or a cfg bit to use as reset 1h INIT (default)
7:4	R/W	6h	T_SATA<0>_CFG_MISC_LINK_SM_MODE: Bit 0 enables holda bypass. Bits 3 and 2 control l_idle features. 6h INIT (default)
3:0	R/W	3h	T_SATA<0>_CFG_MISC_PHY_OOB_SEQ_MODE: bit 0 controls whether we need to back off sending comwake when we receive comwake from drive Bit 1 is used to control our comreset behavior. If this bit is set to 1 then we send only one COMRESET to drive when scontrol_det[0] bit is written. If this bit is 0 then we keep sending COMRESET sequences as long as scontrol_det bit is active. 3h INIT (default)

T_SATA<0>_LOWPOWER_COUNT

Serial ATA Control Register

Offset: 0x554 | Read/Write: R/W | Reset: 0xXXXXXX44

Bit	R/W	Reset	Description
31:8	R	0h	T_SATA<0>_LOWPOWER_COUNT_RSVD: 0h RSVD_31_8
7:4	R/W	4h	T_SATA<0>_LOWPOWER_COUNT_SLUMBER: 4h SLUMBER_INIT (default)
3:0	R/W	4h	T_SATA<0>_LOWPOWER_COUNT_PARTIAL: 4h PARTIAL_INIT (default)

T_SATA<0>_DEVSLP_CTRL0

This register contains various bits to control the PHY

Offset: 0x558 | Read/Write: R | Reset: 0x0000000X

Bit	R/W	Reset	Description
31:1	R	0	Reserved
0	R	None	T_SATA<0>_DEVSLP_CTRL0_PORT0_INTFC_IN_PARTIAL_OR_SLUMBER:

T_SATA<0>_AO_SPARE_0

Spare register on cold reset with zero as the default value

Offset: 0x580 | Read/Write: R/W | Reset: 0x7FFFFFFF

Bit	R/W	Reset	Description
31:0	R/W	7FFFFFFFh	T_SATA<0>_AO_SPARE_0_RSVD: 7FFFFFFFh RSVD_DEFAULT (default)

T_SATA<0>_AO_SPARE_1

Spare register on cold reset with zero as the default value

Offset: 0x584 | Read/Write: R/W | Reset: 0x7FFFFFFF

Bit	R/W	Reset	Description
31:0	R/W	7FFFFFFFh	T_SATA<0>_AO_SPARE_1_RSVD: 7FFFFFFFh RSVD_DEFAULT (default)

T_SATA<0>_INDEX

Index Mask Register

This register implements mask to select writes to channel specific registers. A write to any CHX field in this manual will affect corresponding register for channel n if the nth bit is set in this mask register. If mask register has no bits set then the write will not change any register.

A read from a CHX field will return data corresponding to the channel which has its mask set. If more than one bits are set in mask, then the returned data will be logical OR of register values corresponding to the channels which have mask bits set. If no mask bit is set then the returned value will be 0x00.

The following comment is no longer valid, separate registers are implemented for GEN[1,2,3] Further PEAK and AMP fields can be set differently for gen1 and gen2. A write to these fields will affect the setting for Genx if the bit corresponding to Genx is set in this register.

Offset: 0x680 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:4	R	0	Reserved
3	R/W	0	T_SATA<0>_INDEX_CH4: 0h CH4_UNSELECTED (default) 1h CH4_SELECTED
2	R/W	0	T_SATA<0>_INDEX_CH3: 0h CH3_UNSELECTED (default) 1h CH3_SELECTED
1	R/W	0	T_SATA<0>_INDEX_CH2: 0h CH2_UNSELECTED (default) 1h CH2_SELECTED
0	R/W	0	T_SATA<0>_INDEX_CH1: 0h CH1_UNSELECTED (default) 1h CH1_SELECTED

T_SATA<0>_CHX_MISC

SATA Miscellaneous Control Register

Offset: 0x684 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:1	R	0	Reserved
0	R/W	0	T_SATA<0>_CHX_MISC_LED_DISABLE: 0h LED_DISABLE_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL1_GEN1

SATA PHY Control Register (GEN1)

Offset: 0x690 | Read/Write: R/W | Reset: 0x0000040C

Bit	R/W	Reset	Description
31:28	R	0	Reserved
27:24	R/W	0	T_SATA<0>_CHX_PHY_CTRL1_GEN1_TX_DRV_CNTL: NOTE: this field is reset by Cold Reset 0h TX_DRV_CNTL_DEFAULT (default)
23:20	R/W	0	T_SATA<0>_CHX_PHY_CTRL1_GEN1_TX_PEAK_PRE: NOTE: this field is reset by Cold Reset 0h TX_PEAK_PRE_DEFAULT (default)
19:16	R/W	0	T_SATA<0>_CHX_PHY_CTRL1_GEN1_TX_CMADJ: NOTE: this field is reset by Cold Reset 0h TX_CMADJ_DEFAULT (default)
15:8	R/W	04h	T_SATA<0>_CHX_PHY_CTRL1_GEN1_TX_PEAK: This field contains the PEAK values for GEN1. NOTE: this field is reset by Cold Reset 4h TX_PEAK_DEFAULT (default) 4h TX_PEAK_DEFAULT2 7h TX_PEAK_DEFAULT3 Eh TX_PEAK_DEFAULT4 Eh TX_PEAK_ESATA

Bit	R/W	Reset	Description
7:0	R/W	0Ch	T_SATA<0>_CHX_PHY_CTRL1_GEN1_TX_AMP: This field contains the AMP values for GEN1 This register may have different values based on port usage (esata,gen1,gen) NOTE: this field is reset by Cold Reset Ch TX_AMP_DEFAULT (default) Eh TX_AMP_DEFAULT2 10h TX_AMP_DEFAULT3 14h TX_AMP_DEFAULT4 14h TX_AMP_ESATA

T_SATA<0>_CHX_PHY_CTRL1_GEN2

SATA PHY Control Register (GEN2)

Offset: 0x694 | Read/Write: R/W | Reset: 0x0000A00E

Bit	R/W	Reset	Description
31:28	R	0	Reserved
27:24	R/W	0	T_SATA<0>_CHX_PHY_CTRL1_GEN2_TX_DRV_CNTL: 0h TX_DRV_CNTL_DEFAULT (default)
23:20	R/W	0	T_SATA<0>_CHX_PHY_CTRL1_GEN2_TX_PEAK_PRE: 0h TX_PEAK_PRE_DEFAULT (default)
19:12	R/W	0Ah	T_SATA<0>_CHX_PHY_CTRL1_GEN2_TX_PEAK: This field contains the PEAK values for gen1. Ah TX_PEAK_DEFAULT (default) Ah TX_PEAK_DEFAULT2 Ah TX_PEAK_DEFAULT3 Eh TX_PEAK_DEFAULT4 Eh TX_PEAK_ESATA
11:8	R/W	0	T_SATA<0>_CHX_PHY_CTRL1_GEN2_TX_CMADJ: 0h TX_CMADJ_DEFAULT (default)
7:0	R/W	0Eh	T_SATA<0>_CHX_PHY_CTRL1_GEN2_TX_AMP: This field contains the AMP values for gen1 This register may have different values based on port usage (esata,gen1,gen) Eh TX_AMP_DEFAULT (default) 14h TX_AMP_DEFAULT2 14h TX_AMP_DEFAULT3 1Ah TX_AMP_DEFAULT4 1Ah TX_AMP_ESATA

T_SATA<0>_CHX_PHY_CTRL1_GEN3

SATA PHY Control Register (GEN3)

Offset: 0x698 | Read/Write: R/W | Reset: 0x0000E020

Bit	R/W	Reset	Description
31:28	R	0	Reserved
27:24	R/W	0	T_SATA<0>_CHX_PHY_CTRL1_GEN3_TX_DRV_CNTL: NOTE: this field is reset by Cold Reset 0h TX_DRV_CNTL_DEFAULT (default)
23:20	R/W	0	T_SATA<0>_CHX_PHY_CTRL1_GEN3_TX_PEAK_PRE: NOTE: this field is reset by Cold Reset 0h TX_PEAK_PRE_DEFAULT (default)
19:12	R/W	0Eh	T_SATA<0>_CHX_PHY_CTRL1_GEN3_TX_PEAK: This field contains the PEAK values for gen1. NOTE: this field is reset by Cold Reset Eh TX_PEAK_DEFAULT (default)

Bit	R/W	Reset	Description
			14h TX_PEAK_ESATA
11:8	R/W	0	T_SATA<0>_CHX_PHY_CTRL1_GEN3_TX_CMADJ: NOTE: this field is reset by Cold Reset 0h TX_CMADJ_DEFAULT (default)
7:0	R/W	20h	T_SATA<0>_CHX_PHY_CTRL1_GEN3_TX_AMP: This field contains the AMP values for gen1 This register may have different values based on port usage (esata,gen1,gen) NOTE: this field is reset by Cold Reset 20h TX_AMP_DEFAULT (default) 20h TX_AMP_ESATA

T_SATA<0>_CHX_PHY_CTRL2

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x69c | Read/Write: R/W | Reset: 0x00242424

Bit	R/W	Reset	Description
31:24	R	0	Reserved
23:16	R/W	24h	T_SATA<0>_CHX_PHY_CTRL2_CDR_CNTL_GEN3: 24h CDR_CNTL_GEN3_DEFAULT (default)
15:8	R/W	24h	T_SATA<0>_CHX_PHY_CTRL2_CDR_CNTL_GEN2: 24h CDR_CNTL_GEN2_DEFAULT (default)
7:0	R/W	24h	T_SATA<0>_CHX_PHY_CTRL2_CDR_CNTL_GEN1: 24h CDR_CNTL_GEN1_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL3

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6b0 | Read/Write: R/W | Reset: 0x10004220

Bit	R/W	Reset	Description
31:30	R	0	Reserved
29	R	0	T_SATA<0>_CHX_PHY_CTRL3_STATUS_RX_STAT_IDLE: 0h STATUS_RX_STAT_IDLE_DEFAULT (default)
28	R	1	T_SATA<0>_CHX_PHY_CTRL3_STATUS_TX_STAT_PRESENT: 1h STATUS_TX_STAT_PRESENT_DEFAULT (default)
27:26	R	0	T_SATA<0>_CHX_PHY_CTRL3_STATUS_RX_RATE: 0h STATUS_RX_RATE_DEFAULT (default) 0h STATUS_RX_RATE_GEN1 1h STATUS_RX_RATE_GEN2 2h STATUS_RX_RATE_GEN3
25:24	R	0	T_SATA<0>_CHX_PHY_CTRL3_STATUS_TX_RATE: 0h STATUS_TX_RATE_DEFAULT (default) 0h STATUS_TX_RATE_GEN1 1h STATUS_TX_RATE_GEN2 2h STATUS_TX_RATE_GEN3
23	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_RX_RATE_OVERRIDE: 0h RX_RATE_OVERRIDE_DISABLE (default) 1h RX_RATE_OVERRIDE_ENABLE

Bit	R/W	Reset	Description
22	R	0	Reserved
21:20	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_RX_RATE: 0h RX_RATE_GEN1 (default) 1h RX_RATE_GEN2 2h RX_RATE_GEN3
19	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_TX_RATE_OVERRIDE: 0h TX_RATE_OVERRIDE_DISABLE (default) 1h TX_RATE_OVERRIDE_ENABLE
18	R	0	Reserved
17:16	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_TX_RATE: 0h TX_RATE_GEN1 (default) 1h TX_RATE_GEN2 2h TX_RATE_GEN3
15	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_RX_DATA_READY: 1h RX_DATA_READY_YES 0h RX_DATA_READY_NO (default)
14	R/W	1	T_SATA<0>_CHX_PHY_CTRL3_RX_DATA_EN: 1h RX_DATA_EN_DEFAULT (default)
13	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_TX_DATA_EN_OVERRIDE: 1h TX_DATA_EN_OVERRIDE_YES 0h TX_DATA_EN_OVERRIDE_NO (default)
12	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_TX_DATA_EN: TXDATA_EN and TXDATA_EN_OVERRIDE are used to enable/disable txd_en manually. PHY_CTRL3_TXD_EN and PHY_CTRL3_TXD_EN_OVERRIDE are used to enable/disable txd_en manually. 1h TX_DATA_EN_YES 0h TX_DATA_EN_NO (default)
11	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_RX_SLEEP_OVERRIDE: Enable a forced sleep mode on the SATA port, used for debug of the sleep modes. _NO = Normal Operation _YES = Forced sleep mode Note: The associated SLEEP_MODE register bits select the sleep mode forced. 1h RX_SLEEP_OVERRIDE_YES 0h RX_SLEEP_OVERRIDE_NO (default)
10	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_TX_DATA_READY: 1h TX_DATA_READY_YES 0h TX_DATA_READY_NO (default)
9:8	R/W	2h	T_SATA<0>_CHX_PHY_CTRL3_RX_SLEEP: Select a sleep mode on the SATA port, used for debug of the sleep modes. 00 = Active 01 = Partial Slumber 10 = Slumber 11 = Disabled 2h RX_SLEEP_INIT (default) 0h RX_SLEEP_ACTIVE 1h RX_SLEEP_PARTIAL 2h RX_SLEEP_SLUMBER 3h RX_SLEEP_DISABLED
7	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_TX_SLEEP_OVERRIDE: Enable a forced sleep mode on the SATA port, used for debug of the sleep modes. _NO = Normal Operation _YES = Forced sleep mode Note: The associated SLEEP_MODE register bits select the sleep mode forced. 1h TX_SLEEP_OVERRIDE_YES 0h TX_SLEEP_OVERRIDE_NO (default)
6	R	0	Reserved
5:4	R/W	2h	T_SATA<0>_CHX_PHY_CTRL3_TX_SLEEP: Select a sleep mode on the SATA port, used for debug of the sleep modes. 00 = Active

Bit	R/W	Reset	Description
			01 = Partial Slumber 10 = Slumber 11 = Disabled 2h TX_SLEEP_INIT (default) 0h TX_SLEEP_ACTIVE 1h TX_SLEEP_PARTIAL 2h TX_SLEEP_SLUMBER 3h TX_SLEEP_DISABLED
3	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_CKBUFPD_OVRD: 0h CKBUFPD_OVRD_DEFAULT (default)
2	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_CKBUFPD: 0h CKBUFPD_DEFAULT (default)
1	R	0	Reserved
0	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_IDDQ: 0h IDDQ_INIT (default)

T_SATA<0>_CHX_PHY_CTRL4

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6b4 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:30	R	0	T_SATA<0>_CHX_PHY_CTRL4_SPARE_OUT: 0h SPARE_OUT_DEFAULT (default)
29:28	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_SPARE_IN: 0h SPARE_IN_DEFAULT (default)
27	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_PRBS_CHK_EN: 0h PRBS_CHK_EN_DEFAULT (default)
26	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_TEST_EN: 0h TEST_EN_DEFAULT (default)
25	R	0	Reserved
24	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_RATE_MODE: 0h RATE_MODE_DEFAULT (default)
23:22	R	0	Reserved
21:20	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_RX_DIV: 0h RX_DIV_DEFAULT (default)
19:18	R	0	Reserved
17:16	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_TX_DIV: 0h TX_DIV_DEFAULT (default)
15:14	R	0	Reserved
13	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_RX_CDR_RESET: 0h RX_CDR_RESET_DEFAULT (default)
12	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_TX_SYNC: 0h TX_SYNC_IDLE (default) 1h TX_SYNC_NOW
11	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_FED_LOOP: 0h FED_LOOP_DEFAULT (default)
10:8	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_TX_DATA_MODE: 0h TX_DATA_MODE_NORMAL (default)

Bit	R/W	Reset	Description
			1h TX_DATA_MODE_PRBS_2_7 2h TX_DATA_MODE_0101010101 3h TX_DATA_MODE_1100110011 4h TX_DATA_MODE_0000011111 5h TX_DATA_MODE_0101111100
7	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_FEA_LOOP: 0h FEA_LOOP_DEFAULT (default)
6:4	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_FEA_MODE: 0h FEA_MODE_DEFAULT (default)
3	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_NEA_LOOP: 0h NEA_LOOP_DEFAULT (default)
2	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_NED_LOOP: 0h NED_LOOP_DEFAULT (default)
1:0	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_NED_MODE: 0h NED_MODE_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL5

SATA PHY Control Register

This register contains various bits to control the PHY

Offset: 0x6b8 | Read/Write: R/W | Reset: 0x00000208

Bit	R/W	Reset	Description
31:28	R	0	Reserved
27:24	R/W	0	T_SATA<0>_CHX_PHY_CTRL5_CDR_MODE: 0h CDR_MODE_DEFAULT (default)
23:20	R	0	Reserved
19	R/W	0	T_SATA<0>_CHX_PHY_CTRL5_TX_RDET: 0h TX_RDET_IDLE (default) 1h TX_RDET_START
18	R/W	0	T_SATA<0>_CHX_PHY_CTRL5_RX_IDLE_MODE: 0h RX_IDLE_MODE_DEFAULT (default)
17	R/W	0	T_SATA<0>_CHX_PHY_CTRL5_RX_IDLE_BYP: 0h RX_IDLE_BYP_DISABLE (default) 1h RX_IDLE_BYP_ENABLE
16	R/W	0	T_SATA<0>_CHX_PHY_CTRL5_TX_RDET_BYP: 0h TX_RDET_BYP_DISABLE (default) 1h TX_RDET_BYP_ENABLE
15:14	R/W	0	T_SATA<0>_CHX_PHY_CTRL5_RX_IDLE_T: 0h RX_IDLE_T_DEFAULT (default)
13:12	R/W	0	T_SATA<0>_CHX_PHY_CTRL5_TX_RDET_T: 0h TX_RDET_T_DEFAULT (default)
11:8	R/W	2h	T_SATA<0>_CHX_PHY_CTRL5_TX_SEL_LOAD: 2h TX_SEL_LOAD_DEFAULT (default)
7:4	R	0	Reserved
3:0	R/W	8h	T_SATA<0>_CHX_PHY_CTRL5_MISC_CNTL: 8h MISC_CNTL_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL6

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6bc | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:24	R	0	Reserved
23:16	R	0	T_SATA<0>_CHX_PHY_CTRL6_MISC_OUT: 0h MISC_OUT_DEFAULT (default)
15	R	0	Reserved
14	R	0	T_SATA<0>_CHX_PHY_CTRL6_RX_BYP_IN: 0h RX_BYP_IN_DEFAULT (default)
13	R	0	Reserved
12	R	0	T_SATA<0>_CHX_PHY_CTRL6_TX_BYP_IN: 0h TX_BYP_IN_DEFAULT (default)
11	R	0	Reserved
10	R/W	0	T_SATA<0>_CHX_PHY_CTRL6_RX_BYP_MODE: 0h RX_BYP_MODE_FALSE (default) 1h RX_BYP_MODE_TRUE
9:8	R	0	Reserved
7	R/W	0	T_SATA<0>_CHX_PHY_CTRL6_RX_BYP_EN: 0h RX_BYP_EN_FALSE (default) 1h RX_BYP_EN_TRUE
6	R/W	0	T_SATA<0>_CHX_PHY_CTRL6_RX_BYP_DIR: 0h RX_BYP_DIR_FALSE (default) 1h RX_BYP_DIR_TRUE
5	R	0	Reserved
4	R/W	0	T_SATA<0>_CHX_PHY_CTRL6_RX_BYP_OUT: 0h RX_BYP_OUT_FALSE (default) 1h RX_BYP_OUT_TRUE
3	R/W	0	T_SATA<0>_CHX_PHY_CTRL6_TX_BYP_EN: 0h TX_BYP_EN_FALSE (default) 1h TX_BYP_EN_TRUE
2	R/W	0	T_SATA<0>_CHX_PHY_CTRL6_TX_BYP_DIR: 0h TX_BYP_DIR_FALSE (default) 1h TX_BYP_DIR_TRUE
1	R	0	Reserved
0	R/W	0	T_SATA<0>_CHX_PHY_CTRL6_TX_BYP_OUT: 0h TX_BYP_OUT_FALSE (default) 1h TX_BYP_OUT_TRUE

T_SATA<0>_CHX_PHY_CTRL7

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6c0 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:24	R	0	Reserved

Bit	R/W	Reset	Description
23:20	R/W	0	T_SATA<0>_CHX_PHY_CTRL7_RX_WANDER_GEN3: 0h RX_WANDER_GEN3_DEFAULT (default)
19:18	R/W	0	T_SATA<0>_CHX_PHY_CTRL7_RX_TERM_CNTL_GEN3: 0h RX_TERM_CNTL_GEN3_DEFAULT (default)
17:16	R/W	0	T_SATA<0>_CHX_PHY_CTRL7_TX_TERM_CNTL_GEN3: 0h TX_TERM_CNTL_GEN3_DEFAULT (default)
15:12	R/W	0	T_SATA<0>_CHX_PHY_CTRL7_RX_WANDER_GEN2: 0h RX_WANDER_GEN2_DEFAULT (default)
11:10	R/W	0	T_SATA<0>_CHX_PHY_CTRL7_RX_TERM_CNTL_GEN2: 0h RX_TERM_CNTL_GEN2_DEFAULT (default)
9:8	R/W	0	T_SATA<0>_CHX_PHY_CTRL7_TX_TERM_CNTL_GEN2: 0h TX_TERM_CNTL_GEN2_DEFAULT (default)
7:4	R/W	0	T_SATA<0>_CHX_PHY_CTRL7_RX_WANDER_GEN1: 0h RX_WANDER_GEN1_DEFAULT (default)
3:2	R/W	0	T_SATA<0>_CHX_PHY_CTRL7_RX_TERM_CNTL_GEN1: 0h RX_TERM_CNTL_GEN1_DEFAULT (default)
1:0	R/W	0	T_SATA<0>_CHX_PHY_CTRL7_TX_TERM_CNTL_GEN1: 0h TX_TERM_CNTL_GEN1_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL8

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6c4 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:22	R	0	Reserved
21:16	R	0	T_SATA<0>_CHX_PHY_CTRL8_RX_QEYE_OUT: 0h RX_QEYE_OUT_DEFAULT (default)
15:9	R	0	Reserved
8	R/W	0	T_SATA<0>_CHX_PHY_CTRL8_RX_QEYE_EN_GEN3: 0h RX_QEYE_EN_GEN3_DEFAULT (default)
7:5	R	0	Reserved
4	R/W	0	T_SATA<0>_CHX_PHY_CTRL8_RX_QEYE_EN_GEN2: 0h RX_QEYE_EN_GEN2_DEFAULT (default)
3:1	R	0	Reserved
0	R/W	0	T_SATA<0>_CHX_PHY_CTRL8_RX_QEYE_EN_GEN1: 0h RX_QEYE_EN_GEN1_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL9

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6c8 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:16	R/W	0	T_SATA<0>_CHX_PHY_CTRL9_MISC_TEST:

Bit	R/W	Reset	Description
			0h MISC_TEST_DEFAULT (default)
15:8	R/W	0	T_SATA<0>_CHX_PHY_CTRL9_MISC_OUT_SEL: 0h MISC_OUT_SEL_DEFAULT (default)
7:3	R	0	Reserved
2	R/W	0	T_SATA<0>_CHX_PHY_CTRL9_DFE_RESET: 0h DFE_RESET_DEFAULT (default)
1	R	0	T_SATA<0>_CHX_PHY_CTRL9_DFE_TRAIN_DONE: 0h DFE_TRAIN_DONE_DEFAULT (default)
0	R/W	0	T_SATA<0>_CHX_PHY_CTRL9_DFE_TRAIN_EN: 0h DFE_TRAIN_EN_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL10

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6cc | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:16	R/W	0	T_SATA<0>_CHX_PHY_CTRL10_EOM_CNTL: 0h EOM_CNTL_DEFAULT (default)
15:3	R	0	Reserved
2	R/W	0	T_SATA<0>_CHX_PHY_CTRL10_EOM_EN: 0h EOM_EN_DEFAULT (default)
1	R	0	T_SATA<0>_CHX_PHY_CTRL10_EOM_TRAIN_DONE: 0h EOM_TRAIN_DONE_DEFAULT (default)
0	R/W	0	T_SATA<0>_CHX_PHY_CTRL10_EOM_TRAIN_EN: 0h EOM_TRAIN_EN_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL11

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6d0 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:16	R/W	0	T_SATA<0>_CHX_PHY_CTRL11_GEN2_RX_EQ: 0h GEN2_RX_EQ_DEFAULT (default)
15:0	R/W	0	T_SATA<0>_CHX_PHY_CTRL11_GEN1_RX_EQ: 0h GEN1_RX_EQ_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL12

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6d4 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description

Bit	R/W	Reset	Description
31:16	R	0	Reserved
15:0	R/W	0	T_SATA<0>_CHX_PHY_CTRL12_GEN3_RX_EQ: 0h GEN3_RX_EQ_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL13

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6d8 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:12	R	0	Reserved
11:0	R/W	0	T_SATA<0>_CHX_PHY_CTRL13_CDR_TEST: 0h CDR_TEST_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL14

SATA PHY Control Register

This register contains various bits to control the PHY

Offset: 0x6dc | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	T_SATA<0>_CHX_PHY_CTRL14_DFE_CNTL: 0h DFE_CNTL_DEFAULT (default)

T_SATA<0>_CHXCFG3

Serial ATA Control Register

This register is used to send the SATA bist activate command to the SATA device. The bist code (byte) is loaded then a '1' is written to bit zero.

Offset: 0x700 | Read/Write: R/W | Reset: 0xXXXX0000

Bit	R/W	Reset	Description
31:16	R	None	T_SATA<0>_CHXCFG3_CHX_PRBS_ERROR_CNT: 0h CHX_PRBS_ERROR_CNT_0
15:8	R/W	0	T_SATA<0>_CHXCFG3_CHX_BIST_CODE: NOTE: this field is reset by Cold Reset 0h CHX_BIST_CODE_DEFAULT (default)
7:2	R	0	Reserved
1	R	0	T_SATA<0>_CHXCFG3_CHX_BIST_STAT: NOTE: this field is reset by Cold Reset 1h CHX_BIST_STAT_BUSY 0h CHX_BIST_STAT_NOT_BUSY (default)
0	RW1C	0	T_SATA<0>_CHXCFG3_CHX_BIST_SEND: NOTE: this field is reset by Cold Reset 1h CHX_BIST_SEND_NOW 0h CHX_BIST_SEND_INIT (default)

T_SATA<0>_CHXCFG4_CHX

Offset: 0x704 | Read/Write: R/W | Reset: 0x0000000a

Bit	R/W	Reset	Description
31:13	R	0	Reserved
12	R/W	0	T_SATA<0>_CHXCFG4_CHX_SW_COMWAKE_PI: NOTE: this field is reset by Cold Reset 0h SW_COMWAKE_PI_INIT (default)
11:8	R/W	0	T_SATA<0>_CHXCFG4_CHX_PHY_ALIGN_NUM_CNT: The PHY_ALIGN_NUM_CNT field configures the number of ALIGN primitive pairs that are inserted by the phy interface block into the output stream. The field value represents a number which is one less than the number of ALIGN pairs to send. So, for eg, a value of 0 (default) would send 1 ALIGN pair (i.e 2 ALIGNs) every PHY_ALIGN_DWORD_CNT DWORDs. In other words, every PHY_ALIGN_DWORD_CNT DWORDs, the phy-interface sends out (PHY_ALIGN_NUM_CNT+1)*2 ALIGNs NOTE: this field is reset by Cold Reset 0h PHY_ALIGN_NUM_CNT_INIT (default)
7:0	R/W	0Ah	T_SATA<0>_CHXCFG4_CHX_PHY_ALIGN_DWORD_CNT: The PHY_ALIGN_DWORD_CNT field configures the number of DWORDs sent before the required (PHY_ALIGN_NUM_CNT + 1) number of ALIGN primitive pairs are inserted by the phy-interface block into the output stream. For example, the default PHY_ALIGN_DWORD_CNT setting of 254 and default PHY_ALIGN_NUM_CNT of 0 implies an output stream of 254 DWORDs, 2 ALIGNs, 254 DWORDs, 2 ALIGNs, and so on. To disable insertion of ALIGN primitives, set this field to zero. NOTE: this field is reset by Cold Reset Ah PHY_ALIGN_DWORD_CNT_INIT (default)

T_SATA<0>_PRBS_CHX

This register contains control bits for the IOBIST PRBS generator and checker. Setting PI_LOOPBACK will select the PRBS as the TX data source. Setting PRBS_EN will cause the PRBS to start running. PRBS_SEED contains the 10-bit initial value for the PRBS.

Offset: 0x714 | Read/Write: R/W | Reset: 0xXXXXX001

Bit	R/W	Reset	Description
31:16	R	None	T_SATA<0>_PRBS_CHX_ERROR_COUNT:
15	R	None	T_SATA<0>_PRBS_CHX_LOCKED:
14	R	None	T_SATA<0>_PRBS_CHX_ERROR:
13	R	0	Reserved
12	R/W	1	T_SATA<0>_PRBS_CHX_HOT_RESET: NOTE: this field is reset by Cold Reset 1h HOT_RESET_DEFAULT (default)
11	R	0h	T_SATA<0>_PRBS_CHX_RSVD: 0h RSVD_DEFAULT (default)
10	R/W	0	T_SATA<0>_PRBS_CHX_PI_LOOPBACK: NOTE: this field is reset by Cold Reset 0h PI_LOOPBACK_DEFAULT (default)
9:0	R/W	1	T_SATA<0>_PRBS_CHX_SEED: NOTE: this field is reset by Cold Reset 1h SEED_DEFAULT (default)

T_SATA<0>_CHX_LINK0

Offset: 0x750 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:3	R	0h	T_SATA<0>_CHX_LINK0_RSVD_0: 0h RSVD_0_VAL
2	R/W	1	T_SATA<0>_CHX_LINK0_LANE_IDDQ_ON_OFFLINE: SW writes this bit for iddq NOTE: this field is reset by Cold Reset 1h LANE_IDDQ_ON_OFFLINE_YES (default) 0h LANE_IDDQ_ON_OFFLINE_NO
1	R/W	1	T_SATA<0>_CHX_LINK0_CONT_DISABLE: CONT_DISABLE is used to disable using CONT primitive in the SATA tx. This bit is provided for each channel. NOTE: this field is reset by Cold Reset 0h CONT_DISABLE_NO 1h CONT_DISABLE_YES 1h CONT_DISABLE_DEFAULT (default)
0	R/W	0	T_SATA<0>_CHX_LINK0_SCRAM_DIS: SCRAM_DIS is used to disable data scrambling in the SATA Link layer. This bit is provided for each channel. NOTE: this field is reset by Cold Reset 0h SCRAM_DIS_NO 1h SCRAM_DIS_YES 0h SCRAM_DIS_DEFAULT (default)

T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR

This register is the backdoor register for PXTFD of the PSM registers . Used for the purpose of Power Ungating - SAVE and RESTORE vis BKDOOR . Need to write CHX_INDEX to select the channel specific register

Offset: 0x790 | Read/Write: R/W | Reset: 0xXXXX007F

Bit	Reset	R/W	Description
31:16	0h	R	T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR_RSVD: 0h RSVD_00
15:8	0	R/W	T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR_ERR: NOTE: this field is reset by Cold Reset 0h ERR_00 (default)
7	0	R/W	T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR_STS_BSY: 0h STS_BSY_CLEAR (default) 1h STS_BSY_SET
6	1	R/W	T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR_STS_DRDY: 0h STS_DRDY_CLEAR 1h STS_DRDY_SET (default)
5	1	R/W	T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR_STS_DF: 0h STS_DF_CLEAR 1h STS_DF_SET (default)
4	1	R/W	T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR_STS_CS: 0h STS_CS_CLEAR 1h STS_CS_SET (default)
3	1	R/W	T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR_STS_DRQ: 0h STS_DRQ_CLEAR 1h STS_DRQ_SET (default)
2:1	3h	R/W	T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR_STS_2_1: 3h STS_2_1_INIT (default)
0	1	R/W	T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR_STS_ERR: 0h STS_ERR_CLEAR 1h STS_ERR_SET (default)

T_SATA<0>_CHX_AHCI_PORT_PXSIG_BKDR

This register is the backdoor register for PXSIG of the PSM registers. Used for the purpose of Power Ungating - SAVE and RESTORE vis BKDOOR. Need to write CHX_INDEX to select the channel specific register.

Offset: 0x794 | Read/Write: R/W | Reset: 0xFFFFFFFF

Bit	R/W	Reset	Description
31:24	R/W	FFh	T_SATA<0>_CHX_AHCI_PORT_PXSIG_BKDR_LBA_HIGH: FFh LBA_HIGH_INIT (default) 0h LBA_HIGH_00
23:16	R/W	FFh	T_SATA<0>_CHX_AHCI_PORT_PXSIG_BKDR_LBA_MID: FFh LBA_MID_INIT (default) 0h LBA_MID_00
15:8	R/W	FFh	T_SATA<0>_CHX_AHCI_PORT_PXSIG_BKDR_LBA_LOW: FFh LBA_LOW_INIT (default) 0h LBA_LOW_00
7:0	R/W	FFh	T_SATA<0>_CHX_AHCI_PORT_PXSIG_BKDR_SECTOR_CNT: FFh SECTOR_CNT_INIT (default) 0h SECTOR_CNT_00

T_SATA<0>_CHX_AHCI_PORT_PXSSTS_BKDR

This register is the backdoor register for PXSSTS_SPD of the PSM registers. Only STS_DPD is restored by SW, else will be driven by the HW. Used for the purpose of Power Ungating - SAVE and RESTORE vis BKDOOR. Need to write CHX_INDEX to select the channel specific register.

Offset: 0x798 | Read/Write: R/W | Reset: 0xXXXXX000

Bit	R/W	Reset	Description
31:12	R	0h	T_SATA<0>_CHX_AHCI_PORT_PXSSTS_BKDR_RSVD_31_12: 0h RSVD_31_12_VAL
11:8	R/W	0	T_SATA<0>_CHX_AHCI_PORT_PXSSTS_BKDR_IPM: 0h IPM_NO_DEV (default) 8h IPM_DEVSLEEP 6h IPM_SLUMBER
7:4	R/W	0	T_SATA<0>_CHX_AHCI_PORT_PXSSTS_BKDR_SPD: 0h SPD_NO_DEV (default) 1h SPD_GEN1 2h SPD_GEN2 3h SPD_GEN3
3:0	R/W	0	T_SATA<0>_CHX_AHCI_PORT_PXSSTS_BKDR_DET: 0h DET_NO_DEV (default)

T_SATA<0>_CHX_GLUE

Offset: 0x7f0 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:1	R	0h	T_SATA<0>_CHX_GLUE_RSVD_0: 0h RSVD_0_VAL
0	R/W	1	T_SATA<0>_CHX_GLUE_ATAPI_BLINK_EN: qualifier for led_active signal NOTE: this field is reset by Cold Reset 0h ATAPI_BLINK_EN_0 1h ATAPI_BLINK_EN_1 1h ATAPI_BLINK_EN_DEFAULT (default)

T_SATA<0>_INTR

Serial ATA Reserved Register

Offset: 0xac0 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31	R	0	T_SATA<0>_INTR_SEC_INTR: SEC_INTR is a status bit for Secondary Interrupt pending. Used for debug and emulation purpose. 1h SEC_INTR_PENDING 0h SEC_INTR_NOT_PENDING (default)
30	R	0	T_SATA<0>_INTR_PRI_INTR: PRI_INTR is a status bit for Primary Interrupt pending. Used for debug and emulation purpose. 1h PRI_INTR_PENDING 0h PRI_INTR_NOT_PENDING (default)
29:0	R	0	Reserved

T_SATA<0>_EMU1

Serial ATA Control Register

Offset: 0xc00 | Read/Write: R/W | Reset: 0x42060F0X

Bit	R/W	Reset	Description
31:28	R/W	4h	T_SATA<0>_EMU1_PHY_UART_TIMEOUT: NOTE: this field is reset by Cold Reset 4h PHY_UART_TIMEOUT_INIT (default)
27	R	0	Reserved
26	R/W	0	T_SATA<0>_EMU1_PHY_USE_RBC1: NOTE: this field is reset by Cold Reset 0h PHY_USE_RBC1_NO (default) 1h PHY_USE_RBC1_YES
25	R/W	1	T_SATA<0>_EMU1_PHY_ABSORB_ALIGN_PRIM: NOTE: this field is reset by Cold Reset 0h PHY_ABSORB_ALIGN_PRIM_NO 1h PHY_ABSORB_ALIGN_PRIM_YES (default)
24	R/W	0	T_SATA<0>_EMU1_PHY_BYPASS_EN: NOTE: this field is reset by Cold Reset 0h PHY_BYPASS_EN_NO (default) 1h PHY_BYPASS_EN_YES
23:16	R/W	06h	T_SATA<0>_EMU1_PHY_UART_SAMPLE: NOTE: this field is reset by Cold Reset 6h PHY_UART_SAMPLE_DEFAULT (default)
15:8	R/W	0Fh	T_SATA<0>_EMU1_PHY_UART_DIV: NOTE: this field is reset by Cold Reset Fh PHY_UART_DIV_DEFAULT (default)
7	R	0	Reserved
6	R/W	0	T_SATA<0>_EMU1_PHY_PM_EN: NOTE: this field is reset by Cold Reset 0h PHY_PM_EN_NO (default) 1h PHY_PM_EN_YES
5	R/W	0	T_SATA<0>_EMU1_PHY_TXCLK_EARLY: 0h PHY_TXCLK_EARLY_NO (default) 1h PHY_TXCLK_EARLY_YES
4	R/W	0	T_SATA<0>_EMU1_PHY_DATA_EARLY:

Bit	R/W	Reset	Description
			NOTE: this field is reset by Cold Reset 0h PHY_DATA_EARLY_NO (default) 1h PHY_DATA_EARLY_YES
3:1	R	Unknown	T_SATA<0>_EMU1_PHY_SEL: 0h PHY_SEL_NOTHING 1h PHY_SEL_MARVELL 2h PHY_SEL_SI 3h PHY_SEL_NVDA_EXT 4h PHY_SEL_NVDA_INT
0	R/W	1	T_SATA<0>_EMU1_RESET_ON_COMRESET: NOTE: this field is reset by Cold Reset 1h RESET_ON_COMRESET_YES (default) 0h RESET_ON_COMRESET_NO

T_SATA<0>_EMU2

Serial ATA Backdoor Class Code Register

This register changes the SATA device PCI class code register via a backdoor write. This could be used by SBIOS to update the class code before OS sees it.

Offset: 0xc04 | Read/Write: R/W | Reset: 0x000XXXXX

Bit	R/W	Reset	Description
31	R/W	0	T_SATA<0>_EMU2_RETRY_CTL_FIS_SRST: NOTE: this field is reset by Cold Reset 0h RETRY_CTL_FIS_SRST_INIT (default)
30:27	R/W	0	T_SATA<0>_EMU2_AHCI_DEBUG_PORT_SEL_RESERVED: NOTE: this field is reset by Cold Reset 0h AHCI_DEBUG_PORT_SEL_RESERVED_ZERO (default)
26:24	R/W	0	T_SATA<0>_EMU2_AHCI_DEBUG_PORT_SEL: These bits will be used to select the particular port to debug. It is a zero based number used to select the port. NOTE: this field is reset by Cold Reset 0h AHCI_DEBUG_PORT_SEL_ZERO (default) 1h AHCI_DEBUG_PORT_SEL_ONE 2h AHCI_DEBUG_PORT_SEL_TWO 3h AHCI_DEBUG_PORT_SEL_THREE
23:17	R	0	Reserved
16:0	R	0h	T_SATA<0>_EMU2_RSVD: 0h RSVD_INIT

31.7.5.3 DMA Control Registers

I/O Mapped Registers behind a BAR.

T_SATA_PRI_CMD

Bus Master SATA Primary Channel Command Register

The Primary command register allows host to control the SATA primary channel bus mastering.

Offset: 0x000 | Read/Write: R/W | Reset: 0x0X

Bit	R/W	Reset	Description
7:4	R	0	Reserved
3	R/W	0h	T_SATA_PRI_CMD_BM_RW:

Bit	R/W	Reset	Description
			Read or Write Control. This bit sets the direction of the bus master transfer. When set to zero, PCI bus master reads are performed. When set to one, PCI bus master writes are performed. This bit must NOT be changed when the bus master function is active. 0h BM_RW_READ (default) 1h BM_RW_WRITE
2:1	R	0	Reserved
0	RW1C	None	T_SATA_PRI_CMD_BM_SS: Start/Stop Bus Master. Writing a '1' to this bit enables bus master operation of the controller. Bus master operation begins when this bit is detected changing from a zero to a one. The controller will transfer data between the SATA device and memory only when this bit is set. Master operation can be halted by writing a '0' to this bit. All state information is lost when a '0' is written; Master mode operation cannot be stopped and then resumed. If this bit is reset while bus master operation is still active (i.e., the Bus Master SATA Active bit of the Bus Master SATA Status register for that SATA channel is set) and the drive has not yet finished its data transfer (The Interrupt bit in the Bus Master SATA Status register for that SATA channel is not set), the bus master command is said to be aborted and data transferred from the drive may be discarded before being written to system memory. This bit is intended to be reset after the data transfer is completed, as indicated by either the Bus Master SATA Active bit or the Interrupt bit of the Bus Master SATA Status register for that SATA channel being set, or both. 0h BM_SS_STOP 1h BM_SS_START

T_SATA_PRI_STS

Bus Master SATA Primary Channel Status Register

The Primary channel status registers allows host to check the capabilities and the status of Primary channel bus mastering.

Offset: 0x002 | Read/Write: R/W | Reset: 0xXX

Bit	R/W	Reset	Description
7	R	0h	T_SATA_PRI_STS_SMPLX: Simplex only. This read-only bit is a 0. This indicates that both bus master channels (primary and secondary) can be operated at the same time. 0h SMPLX_NO 1h SMPLX_YES
6	R/W	0h	T_SATA_PRI_STS_DMA1: Drive 1 DMA Capable. This read/write bit is set by device dependent code (BIOS or device driver) to indicate that drive 1 for this channel is capable of DMA transfers, and that the controller has been initialized for optimum performance. 0h DMA1_NO (default) 1h DMA1_YES
5	R/W	0h	T_SATA_PRI_STS_DMA0: Drive 0 DMA Capable. This read/write bit is set by device dependent code (BIOS or device driver) to indicate that drive 0 for this channel is capable of DMA transfers, and that the controller has been initialized for optimum performance. 0h DMA0_NO (default) 1h DMA0_YES
4:3	R	0	Reserved
2	RW1C	0h	T_SATA_PRI_STS_INTR: Interrupt. This bit is set by the rising edge of the SATA interrupt line. This bit is cleared when a '1' is written to it by software. Software can use this bit to determine if an SATA device has asserted its interrupt line. When this bit is read as a one, all data transferred from the drive is visible in system memory. 0h INTR_NO (default) 1h INTR_YES 1h INTR_CLEAR
1	R	0h	T_SATA_PRI_STS_ERR: Error status '0' = No error '1' = Error occured

Bit	R/W	Reset	Description
			0h ERR_NO 1h ERR_YES
0	R	0h	T_SATA_PRI_STS_BMSATAA: Bus Master SATA Active. This bit is set when the Start bit is written to the Command register. This bit is cleared when the last transfer for a region is performed, where EOT for that region is set in the region descriptor. It is also cleared when the Start bit is cleared in the Command register. When this bit is read as a zero, all data transferred from the drive during the previous bus master command is visible in system memory, unless the bus master command was aborted. 0h BMSATAA_NACTV (default) 1h BMSATAA_ACTV

T_SATA_PRI_PRD_ADD

Primary Channel Descriptor Table Pointer Register

Offset: 0x004 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:2	R/W	0h	T_SATA_PRI_PRD_ADD_DW: Base address of Descriptor table. Corresponds to A[31:2] 0h DW_0 (default)
1:0	R	0	Reserved

T_SATA_SEC_CMD

Bus Master SATA Secondary Channel Command Register

The Secondary command register allows host to control the SATA Secondary channel bus mastering.

Offset: 0x008 | Read/Write: R/W | Reset: 0x0X

Bit	R/W	Reset	Description
7:4	R	0	Reserved
3	R/W	0h	T_SATA_SEC_CMD_BM_RW: Read or Write Control. This bit sets the direction of the bus master transfer. When set to zero, PCI bus master reads are performed. When set to one, PCI bus master writes are performed. This bit must NOT be changed when the bus master function is active. 0h BM_RW_READ (default) 1h BM_RW_WRITE
2:1	R	0	Reserved
0	RW1C	None	T_SATA_SEC_CMD_BM_SS: Start/Stop Bus Master. Writing a '1' to this bit enables bus master operation of the controller. Bus master operation begins when this bit is detected changing from a zero to a one. The controller will transfer data between the SATA device and memory only when this bit is set. Master operation can be halted by writing a '0' to this bit. All state information is lost when a '0' is written; Master mode operation cannot be stopped and then resumed. If this bit is reset while bus master operation is still active (i.e., the Bus Master SATA Active bit of the Bus Master SATA Status register for that SATA channel is set) and the drive has not yet finished its data transfer (The Interrupt bit in the Bus Master SATA Status register for that SATA channel is not set), the bus master command is said to be aborted and data transferred from the drive may be discarded before being written to system memory. This bit is intended to be reset after the data transfer is completed, as indicated by either the Bus Master SATA Active bit or the Interrupt bit of the Bus Master SATA Status register for that SATA channel being set, or both. 0h BM_SS_STOP 1h BM_SS_START

T_SATA_SEC_STS

Bus Master SATA Secondary Channel Status Register

The Secondary channel status registers allows host to check the capabilities and the status of Secondary channel Bus mastering

Offset: 0x00a | Read/Write: R/W | Reset: 0xXX

Bit	R/W	Reset	Description
7	R	0h	T_SATA_SEC_STS_SMPLX: Simplex only. This read-only bit is a 0. This indicates that both bus master channels (primary and secondary) can be operated at the same time. 0h SMPLX_NO 1h SMPLX_YES
6	R/W	0h	T_SATA_SEC_STS_DMA1: Drive 1 DMA Capable. This read/write bit is set by device dependent code (BIOS or device driver) to indicate that drive 1 for this channel is capable of DMA transfers, and that the controller has been initialized for optimum performance. 0h DMA1_NO (default) 1h DMA1_YES
5	R/W	0h	T_SATA_SEC_STS_DMA0: Drive 0 DMA Capable. This read/write bit is set by device dependent code (BIOS or device driver) to indicate that drive 0 for this channel is capable of DMA transfers, and that the controller has been initialized for optimum performance. 0h DMA0_NO (default) 1h DMA0_YES
4:3	R	0	Reserved
2	RW1C	0h	T_SATA_SEC_STS_INTR: Interrupt. This bit is set by the rising edge of the SATA interrupt line. This bit is cleared when a '1' is written to it by software. Software can use this bit to determine if an SATA device has asserted its interrupt line. When this bit is read as a one, all data transferred from the drive is visible in system memory. 0h INTR_NO (default) 1h INTR_YES 1h INTR_CLEAR
1	R	0h	T_SATA_SEC_STS_ERR: Error status '0' = No error '1' = Error occurred 0h ERR_NO 1h ERR_YES
0	R	0h	T_SATA_SEC_STS_BMSATAA: Bus Master SATA Active. This bit is set when the Start bit is written to the Command register. This bit is cleared when the last transfer for a region is performed, where EOT for that region is set in the region descriptor. It is also cleared when the Start bit is cleared in the Command register. When this bit is read as a zero, all data transferred from the drive during the previous bus master command is visible in system memory, unless the bus master command was aborted. 0h BMSATAA_NACTV (default) 1h BMSATAA_ACTV

T_SATA_SEC_PRD_ADD

Secondary Channel Descriptor Table Pointer Register

Secondary channel PRD table address provides a pointer to base Physical Resource Descriptor Table in the memory. The table is doubleword aligned and must not cross a 64K byte boundary. Also addresses and count in the PRD table are assumed to be aligned are even numbers.

Offset: 0x00c | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:2	R/W	0h	T_SATA_SEC_PRD_ADD_DW: Base address of Descriptor table. Corresponds to A[31:2] 0h DW_0 (default)

Bit	R/W	Reset	Description
1:0	R	0	Reserved

T_SATA_PRI_COMMAND_0

Offset: 0x1f0 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:24	R/W	0h	T_SATA_PRI_COMMAND_0_LBA_LOW: 0h LBA_LOW_INIT (default)
23:16	R/W	0h	T_SATA_PRI_COMMAND_0_SECTOR_COUNT: 0h SECTOR_COUNT_INIT (default)
15:8	R/W	0h	T_SATA_PRI_COMMAND_0_FEATURE_ERR: 0h FEATURE_ERR_INIT (default)
7:0	R/W	0h	T_SATA_PRI_COMMAND_0_DATA: 0h DATA_INIT (default)

T_SATA_PRI_COMMAND_1

Offset: 0x1f4 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:24	R/W	0h	T_SATA_PRI_COMMAND_1_COMMAND_STATUS: 0h COMMAND_STATUS_INIT (default)
23:16	R/W	0h	T_SATA_PRI_COMMAND_1_DEVICE: 0h DEVICE_INIT (default)
15:8	R/W	0h	T_SATA_PRI_COMMAND_1_LBA_HIGH: 0h LBA_HIGH_INIT (default)
7:0	R/W	0h	T_SATA_PRI_COMMAND_1_LBA_MID: 0h LBA_MID_INIT (default)

T_SATA_PRI_CONTROL

Offset: 0x3f4 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:24	R	0	Reserved
23:16	R/W	0h	T_SATA_PRI_CONTROL_DEVICE_ALTERNATE_STATUS: 0h DEVICE_ALTERNATE_STATUS_INIT (default)
15:0	R	0	Reserved

T_SATA_SEC_COMMAND_0

Offset: 0x170 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:24	R/W	0h	T_SATA_SEC_COMMAND_0_LBA_LOW: 0h LBA_LOW_INIT (default)
23:16	R/W	0h	T_SATA_SEC_COMMAND_0_SECTOR_COUNT: 0h SECTOR_COUNT_INIT (default)
15:8	R/W	0h	T_SATA_SEC_COMMAND_0_FEATURE_ERR: 0h FEATURE_ERR_INIT (default)

Bit	R/W	Reset	Description
7:0	R/W	0h	T_SATA_SEC_COMMAND_0_DATA: 0h DATA_INIT (default)

T_SATA_SEC_COMMAND_1

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

Bit	R/W	Reset	Description
31:24	R/W	0h	T_SATA_SEC_COMMAND_1_COMMAND_STATUS: 0h COMMAND_STATUS_INIT (default)
23:16	R/W	0h	T_SATA_SEC_COMMAND_1_DEVICE: 0h DEVICE_INIT (default)
15:8	R/W	0h	T_SATA_SEC_COMMAND_1_LBA_HIGH: 0h LBA_HIGH_INIT (default)
7:0	R/W	0h	T_SATA_SEC_COMMAND_1_LBA_MID: 0h LBA_MID_INIT (default)

T_SATA_SEC_CONTROL

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

Bit	R/W	Reset	Description
31:24	R	0	Reserved
23:16	R/W	0h	T_SATA_SEC_CONTROL_DEVICE_ALTERNATE_STATUS: 0h DEVICE_ALTERNATE_STATUS_INIT (default)
15:0	R	0	Reserved

31.7.6 SATA Aux Registers

31.7.6.1 SATA_AUX_MISC_CNTL_1_0

MISC_CNTL_1 Register

Offset: 0x1108 | Read/Write: R/W | Reset: 0x00016XX0 (0bxxxxxxxxxxxxxx0010110xxx00xx00000)

Bit	R/W	Reset	Description
18	RW	0x0	AUX_RX_IDLE_STATUS_MASK: Set this bit to mask the aux_rx_idle_status input to the SATA core to 0. 0 = Do not mask the rx_stat_idle input to the SATA core 1 = Mask the rx_stat_idle input to the SATA core to 0. 0 = DISABLE 1 = ENABLE
18	RW	0x0	AUX_OR_CORE_IDLE_STATUS_SEL: This bit is used to select the rx_idle status from either AUX or CORE for interrupt generation. 0 = Take the AUX idle status for interrupt generation 1 = Take the CORE idle status for interrupt generation
17	RW	0x0	DEVSLP_OVERRIDE: Set this bit to override the DEVSLP output from the SATA core. 0 = Assertion of DEVSLP is controlled by the SATA core 1 = DEVSLP is asserted irrespective of the state of DEVSLP from the SATA core. 0 = DISABLE 1 = ENABLE
16	RW	0x1	DSP_SUPPORT: Set this bit to set the AHCI's PxDEVSLP.DSP register bit. 0 = Indicates the SATA port does not support device sleep 1 = Indicates the SATA port supports device sleep. 0 = CLEAR 1 = SET
15	RW	0x0	DESO_SUPPORT: Set this bit to set the AHCI's CAP2.DESO register bit. 0 = Indicates the SATA core has the capability to enter DevSleep from any link state 1 = Indicates the SATA core only has the capability to enter DevSleep from the slumber link state. 0 = CLEAR 1 = SET
14	RW	0x1	SADM_SUPPORT: Set this bit to set the AHCI's CAP2.SADM register bit. 0 = Indicates the SATA core does not have the capability to support hardware assertion of the DEVSLP 1 = Indicates the SATA core has the capability to support hardware assertion of the DEVSLP. 0 = CLEAR 1 = SET
13	RW	0x1	SDS_SUPPORT: Set this bit to set the AHCI's CAP2.SAS register bit. 0 = Indicates the SATA core does not support device sleep 1 = Indicates the SATA core supports device sleep. 0 = CLEAR 1 = SET

Bit	R/W	Reset	Description
12	RW	0x0	RX_STAT_IDLE_MASK: Set this bit to mask the rx_stat_idle input to the SATA core to 0. 0 = Do not mask the rx_stat_idle input to the SATA core 1 = Mask the rx_stat_idle input to the SATA core to 0. 0 = DISABLE 1 = ENABLE
11	RO	X	SATA2IPSM_DEVSLP: Indicates if the SATA link is in DEVSLP. Used for debug purposes.
10:9	RO	X	SATA2IPSM_ST: Indicates whether the SATA link is in partial/slumber modes. Used for debug purposes.
8	RW	0x0	NVA2SATA_OOB_ON_SCONTROL_SPD_WR: If this bit is set to 1, the SATA controller will do an 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 or not SATA controllers do an OOB sequence automatically when they come out of reset. 0 = NO 1 = YES
6:5	RO	X	L0_RX_IDLE_T_SAX: l0_rx_idle_t value from the SATA controller When the SAX partition is power-gated, this field shows a clamped value. 0 = NORMAL
4:3	RW	0x0	L0_RX_IDLE_T_NPG: Sets the l0_rx_idle_t value for the SATA PHY from apb_misc (non-power-gated logic). Before the 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 ensures that the SATA PHY is still receiving the correct threshold setting for OOB detection when the SAX is power-gated. After SAX power is restored, the same value is restored to the l0_rx_idle_t register in the 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 the 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 the external trigger and the 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

31.7.6.2 SATA_AUX_RX_STAT_INT_0

RX_STAT_INT Register

Offset: 0x110c | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxx0xx0xx0xx)

Bit	R/W	Reset	Description
8	RW	0x0	SATA_DEVSLP_INT_DISABLE: SATA devslp interrupt disable 1 = Disables interrupt assertion when devslp interrupt status is set 0 = Enables interrupt assertion when devslp interrupt status is set 0 = ENABLE

Bit	R/W	Reset	Description
			1 = DISABLE
7	RO	X	SATA_DEVSLP: SATA devslp state interrupt status 1 = SATA link is in the devslp state 0 = SATA link is not in the devslp state 0 = NO 1 = YES
6	RO	X	SATA_DEVSLP_INT_STATUS: SATA devslp state interrupt status 1 = devslp interrupt is pending 0 = devslp interrupt is not pending 0 = NONE 1 = PENDING
5	RW	0x0	SATA_DEV_ATTEN_INT_DISABLE: SATA device attention interrupt disable 1 = Disable interrupt assertion when dev_atten interrupt status is set 0 = Enable interrupt assertion when dev_atten interrupt status is set 0 = ENABLE 1 = DISABLE
4	RO	X	SATA_DEVICE_ATTENTION: SATA device attention status. 1 : device attention input is asserted 0 : device attention input is cleared. 0 = NO 1 = YES
3	RO	X	SATA_DEV_ATTEN_INT_STATUS: SATA device attention interrupt status from GPIO. 1 = Device attention interrupt is pending 0 = Device attention interrupt is not pending 0 = NONE 1 = PENDING
2	RW	0x0	SATA_RX_STAT_INT_DISABLE: SATA rx_stat interrupt disable 1 = Disables interrupt assertion when rx_stat interrupt status is set 0 = Enables interrupt assertion when rx_stat interrupt status is set 0 = ENABLE 1 = DISABLE
1	RO	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
0	RO	X	SATA_RX_STAT_INT_STATUS: SATA rx_stat interrupt status from the SATA pad 1 = rx_stat interrupt is pending 0 = rx_stat interrupt is not pending 0 = NONE 1 = PENDING

31.7.6.3 SATA_AUX_RX_STAT_SET_0

RX_STAT_SET Register

Offset: 0x1110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	SATA_DEVSLP_INT_SET: SATA devslp interrupt is set (this bit is auto-cleared). Set the interrupt status bit to '1' when register is written. The read back value is always '0'.
1	0x0	SATA_DEV_ATTEN_INT_SET: SATA dev_atten interrupt is set (this bit is auto-cleared). Set the interrupt status bit to '1' when register is written. The read back value is always '0'.
0	0x0	SATA_RX_STAT_INT_SET: SATA rx_stat interrupt is set (this bit is auto-cleared). Set the interrupt status

Bit	Reset	Description
		bit to '1' when register is written. The read back value is always '0'.

31.7.6.4 SATA_AUX_RX_STAT_CLR_0

RX_STAT_CLR Register

Offset: 0x1114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	SATA_DEVSLP_INT_CLR: SATA devslp interrupt is cleared (this bit is auto-cleared). Clear the interrupt status bit to '1' when register is written. The read back value is always '0'.
1	0x0	SATA_DEV_ATTEN_INT_CLR: SATA dev_atten interrupt is cleared (this bit is auto-cleared). Clear the interrupt status bit to '1' when register is written. The read back value is always '0'.
0	0x0	SATA_RX_STAT_INT_CLR: SATA rx_stat interrupt is cleared (this bit is auto-cleared). Clear the interrupt status bit to '1' when register is written. The read back value is always '0'.

31.7.6.5 SATA_AUX_SPARE_CFG0_0

SPARE_CFG0 Register

Offset: 0x1118 | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxx0000000xxxxxxxx)

Bit	R/W	Reset	Description
14	RW	0x0	MDAT_TIMER_AFTER_PG_VALID: Indicates that the MDAT timer value in the above field is to be loaded in the SATA register space.
13:8	RW	0x0	MDAT_TIMER_AFTER_PG: MDAT timer value to be updated by the SW (PEP driver) before power-ungating the SATA controller.
5:0	RO	X	MDAT_TIMER_BEFORE_PG: MDAT timer value loaded from the SATA register space before SATA is power-gated.

31.7.6.6 SATA_AUX_SPARE_CFG1_0

SPARE_CFG1 Register

Offset: 0x111c | Read/Write: R/W | Reset: 0xffff0000 (0b11111111111111110000000000000000)

Bit	Reset	Description
31:0	-65536	SPARE: Spare register bits

31.7.6.7 SATA_AUX_PAD_PLL_CTRL_0_0

SATA PAD PLL Control Register

Offset: 0x1120 | Read/Write: R/W | Reset: 0x3X3X0000 (0bxx11xxx0xx11xxx100000x0000000000)

Bit	R/W	Reset	Description
29:28	RW	0x3	PLL1_REFCLK_NDIV
27	RO	X	PLL1_LOCKDET
24	RW	0x0	PLL1_MODE
21:20	RW	0x3	PLL0_REFCLK_NDIV: Select feedback divider for PLL.

Bit	R/W	Reset	Description
19	RO	X	PLL0_LOCKDET: Status signal indicating whether PLL is locked within the desired resolution. Forced high when the PLL is in test modes enabled by either PLL_BYPASS_EN or PLL_EMULATION_ON. 0 = Not locked 1 = Locked 0 = NOT_LOCKED 1 = LOCKED
16	RW	0x1	PLL0_MODE
15:12	RW	0x0	REFCLK_SEL: 00 = Internal CML reference clock 01 = Internal CMOS reference clock 1x = External reference clock 0 = INT_CML 1 = INT_CMOS 2 = EXT
11	RW	0x0	REFCLK_TERM100
9	RW	0x0	PLL_CKBUFPD_OVRD
8	RW	0x0	PLL_CKBUFPD_M
7	RW	0x0	PLL_CKBUFPD_BL
6	RW	0x0	PLL_CKBUFPD_BR
5	RW	0x0	PLL_CKBUFPD_TL
4	RW	0x0	PLL_CKBUFPD_TR
3	RW	0x0	SPARE_BIT_2: Reserved bit.
2	RW	0x0	PLL_EMULATION_RSTN: Digital reset for clock divider during emulation mode, hold low (reset) and release to High to set divider phase. No effect during normal operation. 0 = Reset hold 1 = Reset released/active 0 = ASSERT 1 = DEASSERT
1	RW	0x0	SPARE_BIT_1: Reserved bit.
0	RW	0x0	SPARE_BIT_0: Reserved bit. 0 = LOW 1 = HIGH

31.7.6.8 SATA_AUX_PAD_PLL_CTRL_1_0

Offset: 0x1124 | Read/Write: R/W | Reset: 0xXX441020 (0bxxxxxxxx010001000x01000000100000)

Bit	R/W	Reset	Description
31:24	RO	X	PLL_MISC_OUT: Reserved.
23:20	RW	0x4	PLL1_CP_CNTL: Charge-pump current control for PLL1.
19:16	RW	0x4	PLL0_CP_CNTL: Charge-pump current control for PLL0.
15	RW	0x0	PLL_BYPASS_EN: Bypass PLL serial output clocks with input reference clock.
13	RW	0x0	PLL_EMULATION_ON: Enable clock bypass for emulation mode.
12	RW	0x1	TCLKOUT_EN: Enable test clock output pads, TSTCLKP/N.
11:8	RW	0x0	TCLKOUT_SEL: Select internal clock source to bring out through the TSTCLKP/N pads.
7	RW	0x0	XDIGCLK4P5_EN: Enable XDIGCLK4P5 clock output to core.
6	RW	0x0	REFCLKBUF_EN: Enable REFCLKBUF clock output to core.

Bit	R/W	Reset	Description
5	RW	0x1	TXCLKREF_EN: Enable TXCLKREF clock to core.
4	RW	0x0	TXCLKREF_SEL: Select the post divider for TXCLKREF clock.
3	RW	0x0	XDIGCLK_EN: Enable XDIGCLK output clock.
2:0	RW	0x0	XDIGCLK_SEL: Select the output frequency of XDIGCLK.

31.7.6.9 SATA_AUX_PAD_PLL_CTRL_2_0

Offset: 0x1128 | Read/Write: R/W | Reset: 0x0000XX80 (0b0000xxxx00000000x0xxxxxx1xx00000)

Bit	R/W	Reset	Description
31:28	RW	0x0	PLL_TEMP_CNTL
23:18	RW	0x0	PLL_BW_CNTL
17:16	RW	0x0	PLL_BGAP_CNTL
15	RO	X	RCAL_DONE: Status signal to indicate calibration status.
14	RW	0x0	RCAL_RESET: Reset the resistor calibration logic.
12:8	RO	X	RCAL_VAL: Setting of current active resistor calibration code
7	RW	0x1	RCAL_BYPASS: Bypass resistor calibration logic.
4:0	RW	0x0	RCAL_CODE: Sets resistor calibration code when logic is bypassed.

31.7.6.10 SATA_AUX_PAD_PLL_CTRL_3_0

Offset: 0x112c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PLL_MISC_CNTL

31.7.6.11 SATA_AUX_PAD_L0_AUX_CTRL_0_0

Offset: 0x1130 | Read/Write: R/W | Reset: 0x00000X00 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	R/W	Reset	Description
9	RO	X	AUX_RX_IDLE_STATUS
8	RO	X	AUX_TX_RDET_STATUS
5	RW	0x0	AUX_HOLD_EN
4	RW	0x0	AUX_RX_IDLE_MODE
3	RW	0x0	AUX_RX_IDLE_EN
2	RW	0x0	AUX_RX_TERM_EN
1	RW	0x0	AUX_TX_RDET_EN
0	RW	0x0	AUX_TX_TERM_EN

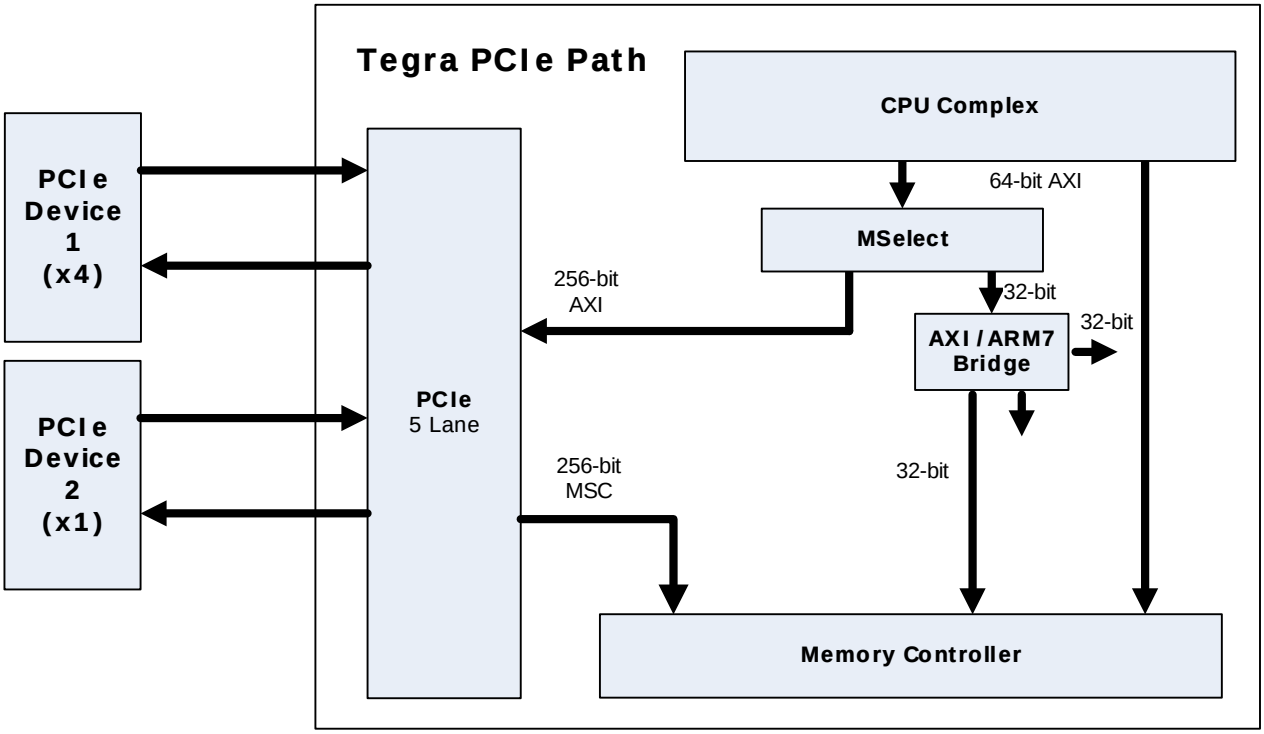
[THIS PAGE INTENTIONALLY LEFT BLANK]

32.0 PCI EXPRESS (PCIe) CONTROLLER

The Tegra® K1 devices incorporate a 5-lane PCIe® root port controller with support for both upstream and downstream AXI interfaces that serves as the control path from Tegra K1 device to the external PCIe device. The PCIe controller enables a connection to one or two PCIe endpoints.

Gen1 (2.5 GT/s) and Gen2 (5.0 GT/s) speeds are supported.

Figure 119: PCIe Interface



32.1 Supported Configurations

The lane mappings for the configurations are internal to the PCIe root port controller. The supported configurations are indicated in the following table.

Use Case	Lane 4	Lane 3	Lane 2	Lane 1	Lane 0
XUSB PADCTL Lane Mapping	Lane 0	Lane 1	Lane 2	Lane 3	Lane 4
Automotive/ Clamshell 1	PCIE x1 (Controller 1)	PCIE x4 (Controller 0)			

32.2 Programming Guidelines

32.2.1 Cold Boot

32.2.1.1 PAD, Clocks and Resets

The AFI clock and the FPCI clocks of the PCIe bridge are sourced from the MSELECT clock, with the clock source and divisor programming determined by MSELECT.

The AFI and PCIe clocks are enabled by setting the following register bits:

- CLK_RST_CONTROLLER_CLK_ ENB_U_SET_0[SET_CLK_ENB_AFI] = 1’b1
- CLK_RST_CONTROLLER_CLK_ ENB_U_SET_0[SET_CLK_ENB_PCIE] = 1’b1

AFI and PCIe resets are deasserted by setting the following register bits:

- CLK_RST_CONTROLLER_RST_DEV_U_CLR_0[CLR_AFI_RST] = 1’b1
- CLK_RST_CONTROLLER_RST_DEV_U_CLR_0[CLR_PCIE_RST] = 1’b1

IOPHYs are brought out of IDDQ by setting the following register bits. Only lanes owned by PCIe should be programmed.

- XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK0] = 1’b1
- XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK1] = 1’b1
- XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK2] = 1’b1
- XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK3] = 1’b1
- XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK4] = 1’b1

32.2.1.2 AFI Initialization

The following register fields should be programmed to match the platform configuration of PCIe:

- AFI_PCIE_CONFIG_0[SM2TMS0_XBAR_CONFIG]
- AFI_PCIE_CONFIG_0[PCIEC1_DISABLE_DEVICE]
- AFI_PCIE_CONFIG_0[PCIEC0_DISABLE_DEVICE]
- AFI_FUSE_0[FUSE_PCIE_T0_GEN2_DIS]

The following register bits should be programmed to match the platform address apertures:

- APBDEV_PMC_GLB_AMAP_CFG_0[PCIE_A3]
- APBDEV_PMC_GLB_AMAP_CFG_0[PCIE_A2]
- APBDEV_PMC_GLB_AMAP_CFG_0[PCIE_A1]

The following register bit should be set to enable FPCI transactions to PCIe:

AFI_CONFIGURATION_0[EN_FPCI] = 1’b1

Refer to the Address Map section in this TRM for Tegra K1 addresses allocated to AFI and PCIe. The following table lists the apertures used by internal AFI and PCIe registers:

Base Address	Size	Usage
0x0100_0000	4KB	PCIe Controller 0 config registers
0x0100_1000	4KB	PCIe Controller 1 config registers
0x0100_2000	256B	PCA 0
0x0100_2100	256B	PCA 1

Base Address	Size	Usage
0x0100_3000	2KB	PCIe Pads
0x0100_3800	2KB	AFI

32.2.1.3 PCIe Root Port Initialization

The following register bit should be set to enable PCIe mastering and accepting memory and I/O requests:

- T_PCIE2_RP_DEV_CTRL_BUS_MASTER = 1'b1
- T_PCIE2_RP_DEV_CTRL_MEMORY_SPACE = 1'b1
- T_PCIE2_RP_DEV_CTRL_IO_SPACE = 1'b1

32.2.1.4 PCIe Hierarchy Enumeration

The configuration register space of the PCIe hierarchy below each PCIe root port is accesses by using the AFI address mapping to convert the Tegra MMIO addresses to PCIe config addresses. The following PCIe registers should be programmed accordingly for access configuration register spaces below the root ports:

- T_PCIE2_RP_BN_LT_SEC_BUS_NUMBER
- T_PCIE2_RP_BN_LT_SUB_BUS_NUMBER

The I/O register space of the PCIe hierarchy below each PCIe root port is accesses by using the AFI address mapping to convert the Tegra MMIO addresses to PCIe I/O addresses. The following PCIe registers should be programmed accordingly for access I/O register spaces below the root ports:

- T_PCIE2_RP_IO_BL_SS_IO_BASE
- T_PCIE2_RP_IO_BL_SS_IO_LIMIT

The I/O register space of the PCIe hierarchy below each PCIe root port is accesses by using the AFI address mapping to shift the Tegra MMIO addresses to PCIe MMIO addresses. The following PCIe registers should be programmed accordingly for access MMIO register spaces below the root ports:

- T_PCIE2_RP_MEM_BL_MEM_BASE
- T_PCIE2_RP_MEM_BL_MEM_LIMIT
- T_PCIE2_RP_PRE_BL_PREFETCH_MEM_BASE
- T_PCIE2_RP_PRE_BL_PREFETCH_MEM_LIMIT

32.2.2 ELPG and LP0

32.2.2.1 Context Save/Restore

Since the PCIe links would be put to L2/L3 when PCIe is put to ELPG or LP0, where the PCIe hierarchy would be reset to exit L2/L3, instead of performing context save/reset, software can reinitialize AFI, PCIe Root Ports, and the PCIe hierarchy after exiting ELPG or LP0.

32.2.2.2 Power Gating Entry

IOPHY are put to IDDQ by setting the following register bits. Only lanes owned by PCIe should be programmed.

- XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK0] = 1'b0
- XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK1] = 1'b0
- XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK2] = 1'b0
- XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK3] = 1'b0

- XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK4] = 1’b0

AFI and PCIe resets are asserted by setting the following register bits:

- CLK_RST_CONTROLLER_RST_DEV_U_SET_0[SET_AFI_RST] = 1’b1
- CLK_RST_CONTROLLER_RST_DEV_U_SET_0[SET_PCIE_RST] = 1’b1

AFI and PCIe clocks are enabled by setting the following register bits:

- CLK_RST_CONTROLLER_CLK_ ENB_U_SET_0[SET_CLK_ENB_AFI] = 1’b1
- CLK_RST_CONTROLLER_CLK_ ENB_U_SET_0[SET_CLK_ENB_PCIE] = 1’b1

AFI and PCIe power is removed by setting the following register fields:

- APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
- APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘PCX’

Then confirm the power removal by reading the following field:

APBDEV_PMC_PWRGATE_STATUS_0[PCX] equals ‘OFF’

32.2.2.3 Power Gating Exit

AFI and PCIe power are restored by setting the following register fields:

- APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
- APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘PCX’

Then confirm the power restore by reading the following field:

APBDEV_PMC_PWRGATE_STATUS_0[PCX] equals ‘ON’

AFI and PCIe clocks are enabled by setting the following register bits:

- CLK_RST_CONTROLLER_CLK_OUT_ENB_U_SET_0[SET_CLK_ENB_AFI] = 1’b1
- CLK_RST_CONTROLLER_CLK_OUT_ENB_U_SET_0[SET_CLK_ENB_PCIE] = 1’b1

AFI and PCIe power clamping are removed by setting the following register field:

APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [PCX]

Then confirm the power clamps’ removal by reading the following field:

APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [PCX] equals ‘0’

AFI and PCIe resets are deasserted by setting the following register bits:

- CLK_RST_CONTROLLER_RST_DEV_U_CLR_0[CLR_AFI_RST] = 1’b1
- CLK_RST_CONTROLLER_RST_DEV_U_CLR_0[CLR_PCIE_RST] = 1’b1

IOPHYs are brought out of IDDQ by setting the following register bits. Only lanes owned by PCIe should be programmed.

- XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK0] = 1’b1
- XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK1] = 1’b1
- XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK2] = 1’b1
- XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK3] = 1’b1
- XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK4] = 1’b1

32.2.2.4 PE_WAKE# in LP0

During LP0, PCIe devices can assert PE_WAKE# to indicate a wake event. Wake from LP0 by asserting PE_WAKE# is enabled by programming the following register fields before entering LP0:

- APBDEV_PMC_WAKE_LVL_0[EVENT[14] to 'ACTIVE_LOW'
- APBDEV_PMC_WAKE_MASK_0[EVENT[14] to 'ENABLE'

After exiting LP0, the PE_WAKE# event should be disabled and the wake status should be cleared by programming the following register fields:

- APBDEV_PMC_WAKE_MASK_0[EVENT[14] to 'DISABLE'
- APBDEV_PMC_WAKE_STATUS_0[EVENT[14] to 1'b1

32.2.2.5 PE_WAKE# in ELPG

When PCIe is under ELPG, PCIe devices can assert PE_WAKE# to signal wake event. Wake from PE_WAKE# assertion when PCIe is under ELPG is enabled by programming the following register fields:

- GPIO_CNF_0.DD[BIT_3] to 'GPIO'
- GPIO_CNF_0.DD[LOCK_3] to 'DISABLE'
- GPIO_OE_0.DD[BIT_3] to 'TRI_STATE'
- GPIO_INT_LVL_0.DD[EDGE_3] to '0'
- GPIO_INT_LVL_0.DD[BIT_3] to 'LOW'
- GPIO_INT_ENB_0.DD[BIT_3] to 'ENABLE'

After exiting LP0, the PE_WAKE# event should be disabled and the wake status should be cleared by programming the following register fields:

- GPIO_INT_ENB_0.DD[BIT_3] to 'DISABLE'
- GPIO_INT_CLR_0.DD[BIT_3] to 'CLEAR'

32.2.3 PLL Power Down

For devices that do not support CLKREQ#, PCIe cannot signal PLL power down during L1. For devices that support CLKREQ#, PCIe should signal PLL power down during L1 if the device deasserts CLKREQ#. PCIe should always signal PLL power down during L2/3.

Whether a device supports CLKREQ# is a platform design time decision. Thus platform software should have the information if CLKREQ# is used. In both cases, software should program the following bits to enable PCIe observing CLKREQ#:

- AFI_PLLE_CONTROL_0[BYPASS_PADS2PLLE_CONTROL] = 1'b0
- AFI_PLLE_CONTROL_0[PADS2PLLE_CONTROL_EN] = 1'b1
- AFI_PEX0_CTRL_0[PEX0_CLKREQ_EN] = 1'b0 for controller 0
- AFI_PEX1_CTRL_0[PEX1_CLKREQ_EN] = 1'b0 for controller 1

32.2.3.1 L1 with Devices Not Supporting CLKREQ#

Software should enable the internal pull-down of the CLKREQ# pin to keep CLKREQ# always asserted if the device does not support CLKREQ#. This will prevent PCIe from signaling PLL power down when link is in L1 because PCIe would consider CLKREQ# always asserted.

- PINMUX_AUX_PEX_L0_CLKREQ_N_0[PUPD] = 'PD'
- PINMUX_AUX_PEX_L1_CLKREQ_N_0[PUPD] = 'PD'

32.2.3.2 L1 with Devices Supporting CLKREQ#

If the platform board does not use an external pull-up on the CLKREQ# pin, software should enable the internal pull-up of the CLKREQ# pin. PCIe will indicate PLL power down when the link is in L1 and CLKREQ# is deasserted.

- PINMUX_AUX_PEX_L0_CLKREQ_N_0[PUPD] = 'PU'
- PINMUX_AUX_PEX_L1_CLKREQ_N_0[PUPD] = 'PU'

32.2.3.3 L2/3

After putting the link to L2/L3, software should program the following bit to override CLKREQ# so PCIe will signal PLL power down irrespective to CLKREQ#.

AFI_PLLE_CONTROL_0[BYPASS_PADS2PLLE_CONTROL] = 1'b1

Software should clear the above bit to 1'b0 before bringing the link back to L0.

32.2.4 MSI and PCIe Messages

32.2.4.1 MSI Handling

Set up MSI Interrupt

Software programs MSI_BAR_SZ to 0x0000F to allocate 64KB to MSI address space.

Software programs MSI_FPCI_BAR_START to 0x00_FEE0_0 to assign the MSI starting address.

For each vector that software programs into the MSI DATA register of the PCIe device, software should also set the corresponding bit in the MSI ENABLE VECTOR[7:0] registers.

Access MSI Vector

Software reads MSI_VECTOR[7:0] registers to identify the bit(s) set to 1 by AFI/IPFS, indicating AFI/IPFS received MSI(s) with matched vector(s), and notify the corresponding interrupt service routine(s).

After software finishes handling the interrupt from the PCIe device, software writes 0 to the vector corresponding bit MSI_VECTOR[7:0] to clear the vector.

32.2.4.2 Message Handling (AFI Only)

Access Message

Software reads the INT_CODE register and determines that the interrupt is for SM_MSG.

Software reads the revised PME register to identify the bit(s) set to 1 by the AFI, indicating the AFI received messages with a matched type.

The INTA/B/C/D PCIe messages and the Root Port Interrupt messages are virtual wires to reflect the status of the interrupts from the source of the interrupts. Thus they are defined as read-only registers to reflect the interrupt status.

If the INTA, INTB, INTC, or INTD bit is set, software should notify the interrupt service routing, where the PCIe devices will send deassert_INTA/B/C/D messages, and the AFI will clear the INTA, INTB, INTC, or INTD bit. If the Root Port Interrupt bits are set, software should notify the interrupt service routing, where the PCIe root ports will send Root_Port_Interrupt_Deassertion messages, and the AFI will clear the Root Port Interrupt bits.

The non-interrupt messages are used for event notifications and reporting. Similar to MSIs, multiple events can be collapsed when there are pending events of the same type (that is, new messages being dropped when the status bits are already set).

If the non-interrupt bits are set, software should handle the messages accordingly, and then write 1s to the bits to clear them.

After the message bit(s) are all cleared, software sets INT_CODE to CLEAR to cause the AFI to deassert the interrupt or move to report the next interrupt source.

32.2.5 PCIe Sideband Signals

32.2.5.1 Removal of PRSNT#

The Tegra K1 does not support dedicated pins for the PCIe PRSNT# pins. To allow presence detection and hot plug support, PCIe should implement software overrides for the PRSNT# inputs of each of controller.

To support presence detection and hot plug, GPIO pins can be allocated to connect to the PRSNT# pins of the PCIe sockets. After being notified that the GPIO pins are pulled to ground as a result of attachment of PCIe cards, software can set the software overrides to signal the event to the PCIe controllers.

32.2.5.2 Control of PE_RST#

Each controller's PE_RST# signal is controlled directly by software programming the following AFI registers:

- Controller 0: AFI_PEX0_CTRL_0[PEX0_RST_L]
- Controller 1: AFI_PEX1_CTRL_0[PEX1_RST_L]

32.2.5.3 Control of CLKREQ# and Reference Clock

The reference clock of each controller is controlled directly by software programming the following AFI registers:

- Controller 0: AFI_PEX0_CTRL_0[PEX0_REFCLK_EN]
- Controller 1: AFI_PEX1_CTRL_0[PEX1_REFCLK_EN]

The reference clock of each controller can be controlled by CLKREQ# by software programming the following AFI registers:

- Controller 0: AFI_PEX0_CTRL_0[PEX0_CLKREQ_EN]
- Controller 1: AFI_PEX1_CTRL_0[PEX1_CLKREQ_EN]

The state of CLKREQ can be read from the following AFI registers:

- Controller 0: AFI_PEX0_STATUS_0[PEX0_CLKREQ_L]
- Controller 1: AFI_PEX1_STATUS_0[PEX1_CLKREQ_L]

Software can enable the AFI interrupt by setting the following bit:

- AFI_AFI_INTR_ENABLE_0[EN_PE_CLKREQ_SENSE]

32.2.6 L1 Clock Gating

Software should set the following register to ensure PCIe can exit from L1:

NV_PROJ__PCIE2_RP_PRIV_MISC_CTLR_CLK_CLAMP_THRESHOLD = 0xF

32.3 PCIe 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

Name	Location	Description
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

32.3.1 PCI Compatible Configuration Registers

32.3.1.1 T_PCIE2_RP_DEV_ID

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: 0x00 | Read/Write: RO | Reset:

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)

32.3.1.2 T_PCIE2_RP_DEV_CTRL

Command and Status Register

Offset: 0x04 | 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
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

Bit	R/W	Reset	Description
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
18:11	R	0	Reserved

Bit	R/W	Reset	Description
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

32.3.1.3 T_PCIE2_RP_REV_CC

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: 0x08 | 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

32.3.1.4 T_PCIE2_RP_MISC_1

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: 0x0c | 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 field is hardwired 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)

32.3.1.5 T_PCIE2_RP_BAR_0

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 hardwired to 0.

Base Address Register 0

Offset: 0x10 | Read/Write: RO

Bit	Reset	Description
31:0	0h	T_PCIE2_RP_BAR_0_RESERVED: 0h: RESERVED_0 (default)

32.3.1.6 T_PCIE2_RP_BAR_1

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 hardwired to 0.

Base Address Register 1

Offset: 0x14 | Read/Write: RO

Bit	Reset	Description
31:0	0h	T_PCIE2_RP_BAR_1_RESERVED: 0h: RESERVED_0 (default)

32.3.1.7 T_PCIE2_RP_BN_LT

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: 0x18 | Read/Write: R/W

Bit	R/W	Reset	Description
31:27	R	0h	T_PCIE2_RP_BN_LT_SLATENCY_TIMER: This field 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: The Subordinate Bus Number field 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 field. The bridge uses this field in conjunction with the Secondary Bus Number field 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 field 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 field. The bridge uses this field 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 field 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 field. The bridge uses this field 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)

32.3.1.8 T_PCIE2_RP_IO_BL_SS

This register implements the I/O Base, I/O Limit, and Secondary Status register as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

The I/O Base and I/O Limit fields 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 field is similar in function and bit definition to the Command and Status register defined above; however, its bits reflect status conditions of the secondary interface (the Command and Status register reflects the status conditions of the primary interface).

I/O Base, I/O Limit and Secondary Status Register

Offset: 0x1c | 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
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

Bit	R/W	Reset	Description
20:16	R	0	Reserved
15:12	R/W	0h	T_PCIE2_RP_IO_BL_SS_IO_LIMIT: This field 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 field 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. Hardwired 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 field 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. Hardwired to 1h. 0h: IO_BASE_SUPPORT_16 1h: IO_BASE_SUPPORT_32 (default)

32.3.1.9 T_PCIE2_RP_MEM_BL

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 fields 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: 0x20 | Read/Write: R/W

Bit	R/W	Reset	Description
31:20	R/W	0h	T_PCIE2_RP_MEM_BL_MEM_LIMIT: This field 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 field must be programmed to a smaller value than the Memory Base field 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 field 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

32.3.1.10 T_PCIE2_RP_PRE_BL

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 fields 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: 0x24 | Read/Write: R/W

Bit	R/W	Reset	Description
31:20	R/W	0h	T_PCIE2_RP_PRE_BL_PREFETCH_MEM_LIMIT: This field 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 field must be programmed to a smaller value than the Prefetchable Memory Base field 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. Hardwired to 1h. 1h: L64BIT_YES (default)
15:4	R/W	1h	T_PCIE2_RP_PRE_BL_PREFETCH_MEM_BASE: This field 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. Hardwired to 1h. 1h: B64BIT_YES (default)

32.3.1.11 T_PCIE2_RP_PRE_BU32

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: 0x28 | 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)

32.3.1.12 T_PCIE2_RP_PRE_LU32

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: 0x2c | 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)

32.3.1.13 T_PCIE2_RP_IO_BL_U16

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 Register

Offset: 0x30 | Read/Write: R/W

Bit	Reset	Description
31:16	0h	T_PCIE2_RP_IO_BL_U16_LIMIT_UPPER_BITS: This field 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 field specifies the upper 16 bits, corresponding to AD[31:16], of the 32-bit I/O base address. 0h: BASE_UPPER_BITS_0 (default)

32.3.1.14 T_PCIE2_RP_CAP_PTR

This register implements the Capabilities Pointer register as defined by the PCI 2.0 Specification.

Capabilities Pointer

Offset: 0x34 | 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)

32.3.1.15 T_PCIE2_RP_ROM_BA

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 hardwired to 0.

Expansion ROM Base Address Register

Offset: 0x38 | Read/Write: RO

Bit	Reset	Description
31:0	0h	T_PCIE2_RP_ROM_BA_RESERVED: 0h: RESERVED_0 (default)

32.3.1.16 T_PCIE2_RP_INTR_BCR

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: 0x3c | 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. Hardwired 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. Hardwired 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. Hardwired to 0. 0h: SECONDARY_DIS_TIMER_LONG (default) 1h: SECONDARY_DIS_TIMER_SHORT
24	R	0h	T_PCIE2_RP_INTR_BCR_PRIMARY_DIS_TIMER: Does not apply to PCI Express. Hardwired 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. Hardwired 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. Hardwired to 0. 0h: MABORT_DISABLED (default) 1h: MABORT_ENABLED

Bit	R/W	Reset	Description
20	R/W	0h	T_PCIE2_RP_INTR_BCR_VGA_16BITIO: When enabled, allows full 16-bit decode of I/O 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 I/O 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 field 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 field. 0h: INTR_PIN_NONE 1h: INTR_PIN_INTA (default) 2h: INTR_PIN_INTB 3h: INTR_PIN_INTC 4h: INTR_PIN_INTD
7:0	R/W	0h	T_PCIE2_RP_INTR_BCR_INTR_LINE: This field contains the interrupt routing information. The field must be implemented by any device (or device function) that uses an interrupt pin. POST software will write the routing information into this field as it initializes and configures the system. The value in this field 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

32.3.2 PCI Subsystem ID and Subsystem Vendor ID Capability Registers

32.3.2.1 T_PCIE2_RP_SS_0

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: 0x40 | 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. Hardwired to 0Dh. Dh: CAP_ID_SS (default)

32.3.2.2 T_PCIE2_RP_SS_1

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: 0x44 | 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 ensure uniqueness (the Vendor ID is used as the SSVID also). 10DEh: SSVID_INIT (default)

32.3.3 PCI Power Management Capability Structure Registers

32.3.3.1 T_PCIE2_RP_PM_0

This register implements the Power Management Capabilities register as defined in Section 7.6 of PCI Express Specifications.

Power Management Capabilities Register

Offset: 0x48 | 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

Bit	Reset	Description
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
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. Hardwired to 01h. 1h: CAP_ID_PM (default)

32.3.3.2 T_PCIE2_RP_PM_1

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: 0x4c | 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. Hardwired 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. Hardwired 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 hardwired 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. Hardwired to 0. 0h: PME_DATA_SCALE_UNS (default)
12:9	R	0h	T_PCIE2_RP_PM_1_PME_DATA_SEL: The PME Data register is not implemented. Hardwired 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

32.3.4 PCI MSI Capability Structure Registers

32.3.4.1 T_PCIE2_RP_MSI_CTRL

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: 0x50 | 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. 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. Hardwired to 05h. 5h: CAP_ID_MSI (default)

32.3.4.2 T_PCIE2_RP_MSI_LOW_ADDR

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: 0x54 | 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)

Bit	R/W	Reset	Description
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 hardwired to 0. 0h: RSVD_0 (default)

32.3.4.3 T_PCIE2_RP_MSI_UPPER_ADDR

The contents of this register specify the upper 32 bits of the 64-bit MSI message address.

MSI Message Upper Address Register

Offset: 0x58 | 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)

32.3.4.4 T_PCIE2_RP_MSI_DATA

The contents of this register specify the data that is written to the MSI message address.

MSI Message Data Register

Offset: 0x5c | 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)

32.3.5 PCI Express Capability Structure Registers

32.3.5.1 T_PCIE2_RP_PCI_EXPRESS_CAPABILITY

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: 0x80 | 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)

Bit	Reset	Description
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)
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)

32.3.5.2 T_PCIE2_RP_DEVICE_CAPABILITY

The Device Capabilities register identifies PCI Express device specific capabilities.

Device Capabilities Register

Offset: 0x84 | 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)

Bit	Reset	Description
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 Specification 1.0a and later incorporated into PCI Express Base Specification 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)
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 µs. • 001b -- 1 µs to less than 2 µs. • 010b -- 2 µs to less than 4 µs. • 011b -- 4 µs to less than 8 us. • 100b – 8 µs to less than 16 µs. • 101b -- 16 µs to less than 32 µs. • 110b -- 32 µs-64 µs. • 111b -- More than 64 µs. 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 µs. • 101b -- 1 µs to less than 2 µs. • 110b -- 2 µs-4µs. • 111b -- More than 4 µs. 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.

Bit	Reset	Description
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)
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 maximum payload size • 001b -- 256B maximum payload size • 010b -- 512B maximum payload size • 011b -- 1024B maximum payload size • 100b -- 2048B maximum payload size • 101b -- 4096B maximum payload size • 110b -- Reserved • 111b -- Reserved 0h: MAX_PAYLOAD_SIZE_INIT (default)

32.3.5.3 T_PCIE2_RP_DEVICE_CONTROL_STATUS

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: 0x88 | 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

Bit	R/W	Reset	Description
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
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 maximum payload size 001b -- 256B maximum payload size 010b -- 512B maximum payload size 011b -- 1024B maximum payload size 100b -- 2048B maximum payload size 101b -- 4096B maximum 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)

Bit	R/W	Reset	Description
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)
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 maximum payload size • 001b -- 256B maximum payload size • 010b -- 512B maximum payload size • 011b -- 1024B maximum payload size • 100b -- 2048B maximum payload size • 101b -- 4096B maximum 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)

32.3.5.4 T_PCIE2_RP_LINK_CAPABILITIES

The Link Capabilities Register identifies PCI Express Link specific capabilities.

Link Capabilities Register

Offset: 0x8c | 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)
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 µs. • 001b -- 1 µs to less than 2 µs. • 010b -- 2 µs to less than 4 µs. • 011b -- 4 µs to less than 8 µs. • 100b -- 8 µs to less than 16 µs. • 101b -- 16 µs to less than 32 µs. • 110b -- 32 µs-64 µs. • 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 µs. • 101b -- 1 µs to less than 2 µs. • 110b -- 2 µs-4 µs. • 111b -- Reserved. 3h: L0S_EXIT_LATENCY_INIT (default)

Bit	Reset	Description
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)
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 0010b = 5.0 Gb/s Link All other encodings are reserved. 2h: MAX_LINK_SPEED_INIT (default)

32.3.5.5 T_PCIE2_RP_LINK_CONTROL_STATUS

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: 0x90 | 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.

Bit	R/W	Reset	Description
			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.
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:

Bit	R/W	Reset	Description
			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)
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)

32.3.5.6 T_PCIE2_RP_SLOT_CAPABILITIES

The Slot Capabilities register identifies PCI Express slot specific capabilities.

Slot Capabilities Register

Offset: 0x94 | 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 field 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 field 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.

Bit	Reset	Description
		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: 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)

32.3.5.7 T_PCIE2_RP_SLOT_CONTROL_STATUS

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: 0x98 | 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, that is, 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 field 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 field 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 bit 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.

Bit	R/W	Reset	Description
			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
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 bit 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 bit return the current state of the Power Indicator; writes to this bit set the Power Indicator. 00b = Reserved 01b = On 10b = Blink 11b = Off Writes to this bit 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 field return the current state of the Attention Indicator; writes to this field set the Attention Indicator. Defined encodings are: 00b = Reserved 01b = On 10b = Blink 11b = Off Writes to this field 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)

Bit	R/W	Reset	Description
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 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)

32.3.5.8 T_PCIE2_RP_RCR

The Root Control Register controls PCI Express Root Complex specific parameters. Bit positions 31:4 of this address are reserved.

Root Control Register

Offset: 0x9c | 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)

32.3.5.9 T_PCIE2_RP_RSR

The Root Status Register provides information about PCI Express device specific parameters. Bit positions 31:18 are reserved.

Root Status Register

Offset: 0xA0 | Read/Write: R/W

Bit	R/W	Reset	Description
31:18	R	0	Reserved
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.

32.3.5.10 T_PCIE2_RP_DEVICE_CAPABILITIES_2

Offset: 0xA4 | 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 Root Port. Ranges A and B are currently supported by the PCIe 2.0 design. 3h: CPL_TO_RANGES_SUP_0 (default)

32.3.5.11 T_PCIE2_RP_DEVICE_CONTROL_STATUS_2

This register is primarily used for Root Port Completion Timeout values. Bits 31:5 are reserved.

Offset: 0xA8 | 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_INIT (default)
4	0h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_2_CPL_TO_DISABLE: Disables completion timeout, making the Root Port 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 µs to 100 ms (80 µs 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 µs to 50 ms (10 ms in design) Note: When CPL_TO_OVERRIDE is enabled, the value in this field is ignored, and the value in CPL_TO_CUSTOM_VALUE is used. 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)

32.3.5.12 T_PCIE2_RP_LINK_CAPABILITIES_2

This is a placeholder register and is hardwired to 0. There are no capabilities that require this register.

Offset: 0xac | Read/Write: RO

Bit	Reset	Description
31:0	0h	T_PCIE2_RP_LINK_CAPABILITIES_2_BITS: 0h: BITS_0

32.3.5.13 T_PCIE2_RP_LINK_CONTROL_STATUS_2

Offset: 0xB0 | 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: 2.5GT/s (Gen1) 0010b: 5GT/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)

32.3.5.14 T_PCIE2_RP_SLOT_CAPABILITIES_2

This is a placeholder register and is hardwired to 0. There are no capabilities that require this register.

Offset: 0xB4 | Read/Write: RO

Bit	Reset	Description
31:0	0h	T_PCIE2_RP_SLOT_CAPABILITIES_2_BITS: 0h: BITS_0

32.3.5.15 T_PCIE2_RP_SLOT_CONTROL_STATUS_2

This is a placeholder register and is hardwired to 0. There are no capabilities that require this register.

Offset: 0xB8 | Read/Write: RO

Bit	Reset	Description
31:0	0h	T_PCIE2_RP_SLOT_CONTROL_STATUS_2_BITS: 0h: BITS_0

32.3.6 Error Reporting Capability Registers

32.3.6.1 T_PCIE2_RP_ERPTCAP

This is the Advanced Error Reporting Enhanced Capability Header register.

Offset: 0x100 | 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)

32.3.6.2 T_PCIE2_RP_ERPTCAP_UCERR

This is the Uncorrectable Error Status register.

Offset: 0x104 | 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)

32.3.6.3 T_PCIE2_RP_ERPTCAP_UCERR_MK

This is the Uncorrectable Error Mask register.

Offset: 0x108 | 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)

32.3.6.4 T_PCIE2_RP_ERPTCAP_UCERR_SEVR

This is the Correctable Error Status register.

Offset: 0x10C | 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

32.3.6.5 T_PCIE2_RP_ERPTCAP_CERR

This is the Correctable Error Status register.

Offset: 0x110 | 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

32.3.6.6 T_PCIE2_RP_ERPTCAP_CERR_MK

This is the Correctable Error Mask register.

Offset: 0x114 | 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

32.3.6.7 T_PCIE2_RP_ERPTCAP_ADV_ERR_CAP_CNTL

This is the Advanced Error Capabilities and Control register.

Offset: 0x118 | 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:

32.3.6.8 T_PCIE2_RP_ERPTCAP_HDR_LOG_DW0

This register is 16 bytes wide. The header is captured such that the fields of the header read by software in the same way the headers are presented in the specification, 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: 0x11C | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	None	T_PCIE2_RP_ERPTCAP_HDR_LOG_DW0_0:

32.3.6.9 T_PCIE2_RP_ERPTCAP_HDR_LOG_DW1

Offset: 0x120 | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	None	T_PCIE2_RP_ERPTCAP_HDR_LOG_DW0_1:

32.3.6.10 T_PCIE2_RP_ERPTCAP_HDR_LOG_DW2

Offset: 0x124 | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	None	T_PCIE2_RP_ERPTCAP_HDR_LOG_DW0_2:

32.3.6.11 T_PCIE2_RP_ERPTCAP_HDR_LOG_DW3

Offset: 0x128 | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	None	T_PCIE2_RP_ERPTCAP_HDR_LOG_DW0_3:

32.3.6.12 T_PCIE2_RP_ERPTCAP_ERR_CMD

This is the Root Error Command register.

Offset: 0x12C | 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

32.3.6.13 T_PCIE2_RP_ERPTCAP_ERR_STS

This is the Root Error Status register.

Offset: 0x130 | 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. Hardware 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	0h	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	0h	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	0h	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	0h	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	0h	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

32.3.6.14 T_PCIE2_RP_ERPTCAP_ERR_ID

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: 0x134 | 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)

32.3.6.15 T_PCIE2_RP_PRIV_XP_DL_0

The T_PCIE2_RP_PRIV_XP_DL_0 register controls various configurable registers in the Data Link layer.

Offset: 0x494 | 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 field 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 specification, except that it should include the Rx_L0s_Adjustment. This field 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 field 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 specification, except that it should include the Tx_L0s_Adjustment, and should *not* include the Internal Delay. This field 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 field 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 field 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 field enables the Gen2 DL timer settings to take effect in 5.0GT/s speed of operation. 0h: GEN2_DL_TIMERS_DISABLE_INIT (default)

32.3.6.16 T_PCIE2_RP_PIPE_CTL

This register controls (enabling /disabling) the pipelines added to meet timing on various interfaces. A value of 0 disable the extra pipeline.

Offset: 0xD5C | 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)
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)

Bit	R/W	Reset	Description
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)

32.3.7 Vendor-Defined Registers

32.3.7.1 T_PCIE2_RP_RX_HDR_LIMIT

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: 0xE00 | 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 field 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 field must be halved, and programmed before both ufpci_reset_ and xreset_. This field must be programmed with an even value. 0h: NP_INIT (default)

32.3.7.2 T_PCIE2_RP_RX_DATA_LIMIT

These buffers reside in UBFI. RXL provides the write interface. The buffers constitute asynchronous boundary of xclk -> ufpci_clk.

Offset: 0xE04 | Read/Write: R/W

Bit	R/W	Reset	Description
31:28	R	0	Reserved
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)

Bit	R/W	Reset	Description
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 field 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)

32.3.7.3 T_PCIE2_RP_TX_HDR_LIMIT

These buffers reside in TXBA. DFI provides the write interface. These buffers constitute an asynchronous boundary of dfpci_clk -> xclk.

Offset: 0xE08 | 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 the 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 the same value as TX_DATA_LIMIT_NP. 0h: NP_INIT (default)

32.3.7.4 T_PCIE2_RP_TX_DATA_LIMIT

These buffers reside in TXBA. DFI provides the write interface. These buffers constitute an asynchronous boundary of dfpci_clk -> xclk.

Offset: 0xE0C | 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)

32.3.7.5 T_PCIE2_RP_UFPCI

This register controls upstream FPCI control and arbitration in UBFI and performance fields.

Offset: 0xE10 | 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 posted write has received this number of grants when the priority was swapped to posted write, its 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 a posted write while there are posted writes pending, then swap the priority to the posted write. 0h: PW_STARV_COUNT_INIT (default)

32.3.7.6 T_PCIE2_RP_MISC0

This register controls miscellaneous fields.

Offset: 0xE14 | 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 µs. 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 hardware 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 don't 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 posted write 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 its unitid+1 and an additional 32 source tags when it has exhausted its allotted use of first 32 source tags. 0h: ENABLE_CLUMPING_INIT (default)

32.3.7.7 T_PCIE2_RP_TXBA0

This register is specific to the TXBA sub-unit in PCIe 2.0 and controls:

- Replay buffer limits
- Completion merging limits
- Replay timer control fields

Offset: 0xE18 | 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 hardware calculates. When this bit is 0, the value in TXBA0_REPLAY_TIMER_EXPIRY is selected (if its non-zero) instead of the hardware 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 up to 32 DWs will be merged. Completions that belong to the same read request, but have payloads greater than 32 DWs will have one fragment merged up to 32 DWs, 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 field exists per controller. It decides how much of the replay buffer (which is shared among all the controllers in a TMS) is used for the controller. The 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)

32.3.7.8 T_PCIE2_RP_TXBA1

This register is specific to the TXBA sub-unit in PCIe 2.0.It controls:

- Replay buffer limits
- Completion merging limits
- Replay timer control fields

Offset: 0xE1C | 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	4h	T_PCIE2_RP_TXBA1_CM_OVER_PW_BURST: This field and the PW_OVER_CM_BURST field are input into the downstream arbiter. If there are several downstream posted writes and downstream Completion Merges (CMs), the TXBA arbiter runs PW_OVER_CM_BURST many posted writes before switching to CM. Once the arbiter starts running completions, it runs CM_OVER_PW_BURST completions before switching back to posted writes. 4h: CM_OVER_PW_BURST_INIT (default)
3:0	R	4h	T_PCIE2_RP_TXBA1_PW_OVER_CM_BURST: This field and the CM_OVER_PW_BURST field are inputs into the downstream arbiter. If there are several downstream posted writes and downstream CMs, the TXBA arbiter runs PW_OVER_CM_BURST many posted writes before switching to CM. Once the arbiter starts running completions, it runs CM_OVER_PW_BURST completions before switching back to PW. 4h: PW_OVER_CM_BURST_INIT (default)

32.3.7.9 T_PCIE2_RP_FORCEFC

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: 0xE20 | 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)

32.3.7.10 T_PCIE2_RP_TIMEOUT0

This is the LINK LTSSM Timeout 0 register.

Note: The value of this register is correct only for the default frequency of clk25m. If the clk25m frequency is changed, then software is responsible for changing the value of this register.

Offset: 0xE24 | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R/W	7h	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 clk25m clock periods. Default value is 7 (~500 ns).. 7h: PAD_SPDCHNG_GEN2_INIT (default) 7h: PAD_PWRUP_GEN2_12MHZ_CLK25M 7h: PAD_PWRUP_GEN2_13MHZ_CLK25M 9h: PAD_PWRUP_GEN2_16_8MHZ_CLK25M Ah: PAD_PWRUP_GEN2_19_2MHZ_CLK25M Dh: PAD_PWRUP_GEN2_24MHZ_CLK25M Eh: PAD_PWRUP_GEN2_26MHZ_CLK25M
23:8	R/W	F0h	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 clk25m clock periods. Default value is 0xF0 (~20 µs). F0h: PAD_PWRUP_CM_INIT (default) F0h: PAD_PWRUP_CM_12MHZ_CLK25M 104h: PAD_PWRUP_CM_13MHZ_CLK25M 150h: PAD_PWRUP_CM_16_8MHZ_CLK25M 180h: PAD_PWRUP_CM_19_2MHZ_CLK25M 1E0h: PAD_PWRUP_CM_24MHZ_CLK25M 208h: PAD_PWRUP_CM_26MHZ_CLK25M
7:0	R/W	7h	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 clk25m clock periods (83 ns when using a 12 MHz clk25m clock). Default value is 7 (~500 ns).. 7h: PAD_PWRUP_INIT (default) 7h: PAD_PWRUP_12MHZ_CLK25M 7h: PAD_PWRUP_13MHZ_CLK25M 9h: PAD_PWRUP_16_8MHZ_CLK25M Ah: PAD_PWRUP_19_2MHZ_CLK25M Dh: PAD_PWRUP_24MHZ_CLK25M Eh: PAD_PWRUP_26MHZ_CLK25M

32.3.7.11 T_PCIE2_RP_TIMEOUT1

This is the LINK LTSSM Timeout 1 register.

Offset: 0xE28 | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R/W	49h	T_PCIE2_RP_TIMEOUT1_RCVRY_SPD_UNSUCCESS_EIDLE: Amount of time to wait in E-Idle after an unsuccessful speed change. Measured in units of clk25m clock periods (83 ns when using a 12 MHz clock). Note: This value is correct only for the default frequency of clk25m. If the clk25m frequency is changed, then software is responsible for changing the value of this field. 49h: RCVRY_SPD_UNSUCCESS_EIDLE_INIT (default) 49h: UNSUCCESS_EIDLE_12MHZ_CLK25M 4Eh: UNSUCCESS_EIDLE_13MHZ_CLK25M 65h: UNSUCCESS_EIDLE_16_8MHZ_CLK25M 74h: UNSUCCESS_EIDLE_19_2MHZ_CLK25M 91h: UNSUCCESS_EIDLE_24MHZ_CLK25M 9Dh: UNSUCCESS_EIDLE_26MHZ_CLK25M

Bit	R/W	Reset	Description
23:16	R/W	Ah	T_PCIE2_RP_TIMEOUT1_RCVRY_SPD_SUCCESS_EIDLE: Amount of time to wait in E-Idle after a successful speed change. Measured in units of clk25m clock periods. Note: This value is correct only for the default frequency of clk25m. If the clk25m frequency is changed, then software is responsible for changing the value of this field. Ah: RCVRY_SPD_SUCCESS_EIDLE_INIT (default) Ah: SUCCESS_EIDLE_12MHZ_CLK25M Bh: SUCCESS_EIDLE_13MHZ_CLK25M Eh: SUCCESS_EIDLE_16_8MHZ_CLK25M 10h: SUCCESS_EIDLE_19_2MHZ_CLK25M 14h: SUCCESS_EIDLE_24MHZ_CLK25M 15h: SUCCESS_EIDLE_26MHZ_CLK25M
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 1 µs. 2E8h: PAD_SPDCHNG_GEN1_INIT (default)

32.3.7.12 T_PCIE2_RP_PRBS

This is the PRBS Results register.

Offset: 0xE34 | 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.

32.3.7.13 T_PCIE2_RP_PRBS_ERR_COUNT

This is the PRBS Results register.

Offset: 0xE38 | 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
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)

32.3.7.14 T_PCIE2_RP_EIDLE_INFER_TO_0

This is an EIDLE inference timeout register.

Offset: 0xE3C | 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)

32.3.7.15 T_PCIE2_RP_EIDLE_INFER_TO_1

This is an EIDLE inference timeout register.

Offset: 0xE40 | 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)

32.3.7.16 T_PCIE2_RP_LTSSM_DBGREG

This is the LTSSM debug register.

Offset: 0xE44 | 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
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

Bit	R/W	Reset	Description
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
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

Bit	R/W	Reset	Description
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

32.3.7.17 T_PCIE2_RP_PRIV_ERRSTS

This is the Detailed Private Error status register.

Offset: 0xE48 | 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
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

Bit	R/W	Reset	Description
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
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

Bit	R/W	Reset	Description
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

32.3.7.18 T_PCIE2_RP_PRIV_ERRMSK

This is the Detailed Private Error status register.

Offset: 0xE4C | 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

Bit	R/W	Reset	Description
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
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

Bit	R/W	Reset	Description
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

32.3.7.19 T_PCIE2_RP_LTSSM_TRACE_CONTROL

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: 0xE50 | 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)
7:4	R/W	0h	T_PCIE2_RP_LTSSM_TRACE_CONTROL_TRIG_LTSSM_MAJOR: 0h: TRIG_LTSSM_MAJOR_INIT (default)

Bit	R/W	Reset	Description
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, the RAM is updated every time LTSSM state change happens irrespective of whether the write pointer has wrapped around. When this bit is clear, the RAM update is stopped when the RAM is full. RAM needs 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 needs to be updated whenever an ltssm_state change happens. 1h: STORE_EN_INIT (default) 0h: STORE_EN_CLEAR 1h: STORE_EN_SET

32.3.7.20 T_PCIE2_RP_LTSSM_TRACE_STATUS

This register helps to check the LTSSM trace which has been stored in RAM.

Offset: 0xE54 | 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 field 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 field 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 field 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 field 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 the 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.

32.3.7.21 T_PCIE2_RP_PG

This register helps power gate the PCIe TMS in NVIDIA-specific ways.

Offset: 0xE58 | 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. software 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

32.3.7.22 T_PCIE2_RP_VAR_RANGE0

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: 0xE5C | 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)

32.3.7.23 T_PCIE2_RP_VAR_RANGE1

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: 0xE60 | 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)

32.3.7.24 T_PCIE2_RP_VEND_XP

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: 0xF00 | 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 likelyhood of loosing 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

32.3.7.25 T_PCIE2_RP_VEND_XP1

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: 0xF04 | 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 µs to less than 2 µs. • 010b: 2 µs to less than 4 µs. • 011b: 4 µs to less than 8 µs. • 100b: 8 µs to less than 16 µs. • 101b: 16 µs to less than 32 µs. • 110b: 32 µs-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 I/O and Config, 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

Bit	R/W	Reset	Description
26:19	R/W	0h	T_PCIE2_RP_VEND_XP1_LINK_PVT_CTL: bit[0] : ACK_L1_NO_WAIT Enable behavior described in the Errata C7 of the PCI Express Base Specification, v1.0a, 7 October 2003. Software should always set this bit to 1. When 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 the Tx side drivers should enter L0s or not. If bit[4] == 1 (override), then bit 0 of T_PCIE2_RP_LINK_CONTROL_STATUS is ignored, and Tx.L0s is not entered. bit [7] : NP_REQ_TIMEOUT If set, disable downstream NP request timeout. This is for debug purposes to hang instead of timeout. 0h: LINK_PVT_CTL_INIT (default)
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)

32.3.7.26 T_PCIE2_RP_VEND_XP2

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: 0xF08 | 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 awaken from L0s. 0h: L0S_ACK_WAKE_INIT (default)

32.3.7.27 T_PCIE2_RP_VEND_XV_TIMEOUT

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: 0xF0C | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R/W	0300 00FAh	Reserved

32.3.7.28 T_PCIE2_RP_VEND_SLOT_STRAP

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: 0xF20 | 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

32.3.7.29 T_PCIE2_RP_ECTL_2_R1

Offset: 0xF34 | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R/W	0h	T_PCIE2_RP_ECTL_2_R1_RX_DFE_1C: 0h: R1_RX_DFE_1C_DEFAULT
15:8	R/W	0h	T_PCIE2_RP_ECTL_2_R1_RX_EQ_1C: 0h: R1_RX_EQ_1C_DEFAULT
7:0	R/W	88h	T_PCIE2_RP_ECTL_2_R1_CDR_CNTL_1C: 88h: R1_CDR_CNTL_1C_DEFAULT

32.3.7.30 T_PCIE2_RP_ECTL_3_R1

Offset: 0xF38 | Read/Write: R/W

Bit	R/W	Reset	Description
11:8	R/W	0h	T_PCIE2_RP_ECTL_3_R1_TX_PEAK_PRE_1C: 0h: R1_TX_PEAK_PRE_1C_DEFAULT
4:0	R/W	Ah	T_PCIE2_RP_ECTL_3_R1_TX_PEAK_ 1C: Ah: R1_TX_PEAK_1C_DEFAULT

32.3.7.31 T_PCIE2_RP_ECTL_2_R2

Offset: 0xF3C | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R/W	0h	T_PCIE2_RP_ECTL_2_R2_RX_DFE_1C: 0h: R2_RX_DFE_1C_DEFAULT
15:8	R/W	0h	T_PCIE2_RP_ECTL_2_R2_RX_EQ_1C: 0h: R2_RX_EQ_1C_DEFAULT

Bit	R/W	Reset	Description
7:0	R/W	88h	T_PCIE2_RP_ECTL_2_R2_CDR_CNTL_1C: 88h: R2_CDR_CNTL_1C_DEFAULT

32.3.7.32 T_PCIE2_RP_ECTL_3_R2

Offset: 0xF40 | Read/Write: R/W

Bit	R/W	Reset	Description
27:24	R/W	0h	T_PCIE2_RP_ECTL_3_R2_TX_PEAK_PRE_SEL1_1C: 0h: R2_TX_PEAK_PRE_SEL1_1C_DEFAULT
20:16	R/W	Ah	T_PCIE2_RP_ECTL_3_R2_TX_PEAK_SEL1_1C: Ah: R2_TX_PEAK_SEL1_1C_DEFAULT
11:8	R/W	0h	T_PCIE2_RP_ECTL_3_R2_TX_PEAK_PRE_SEL0_1C: 0h: R2_TX_PEAK_PRE_SEL0_1C_DEFAULT
4:0	R/W	Eh	T_PCIE2_RP_ECTL_3_R2_TX_PEAK_SEL0_1C: Eh: R2_TX_PEAK_SEL0_1C_DEFAULT

32.3.7.33 T_PCIE2_RP_VEND_CTL1

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: 0xF48 | 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
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, the UBFI allows the CPL to bypass posted writes regardless of the 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

Bit	R/W	Reset	Description
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
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)

32.3.7.34 T_PCIE2_RP_VEND_XP_BIST

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

IOBIST and Characterization Control Register

Offset: 0xF4C | Read/Write: R/W

Bit	R/W	Reset	Description
31	R	0	Reserved

Bit	R/W	Reset	Description
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

32.3.7.35 T_PCIE2_RP_VEND_XP_FTS

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: 0xF54 | 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 to 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 to 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)

32.3.7.36 T_PCIE2_RP_VEND_XP_STATS0

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: 0xF58 | 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)
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

Bit	R/W	Reset	Description
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. Note: This counter will count the number of 8b10b errors in a time window of size 14 clk25m clock periods from the time a 1 is written into this field. Depending on the clock frequency of clk25m, this window will be sized differently. Default is 12 clock periods (1 us). 0h: 8B10B_ERRORS_LC_INIT (default)

32.3.7.37 T_PCIE2_RP_VEND_XP_STATS1

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: 0xF5C | 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).

32.3.7.38 T_PCIE2_RP_VEND_ERROR_COUNT

This register contains 8-bit saturating counters for some of RXL and TXBA reported errors.

Offset: 0xF60 | 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 the number of times that TXBA replays. Saturates at 8'hFF. Can be cleared by writing 1 to PRIV_ERRSTS_REPLAY_STARTED_ERR.
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.

32.3.7.39 T_PCIE2_RP_CFG_MISC

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: 0xF64 | 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 cpu_clk and m2clk can be gated, 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 cpu_clk to m2clk. 0h: MUTE_IDLE_MIN (default) FFh: MUTE_IDLE_MAX

32.3.7.40 T_PCIE2_RP_PRIV_XP_INIT_RECOVERY

Each new time frame, the number of 8B/10B errors is counted. As soon as the number of 8B/10B errors is equal to the threshold, the link is sent into recovery. If at the end of the time frame, the number of 8B/10B errors is smaller than the threshold, a new time frame is started in which the 8B/10B errors are counted.

Offset: 0xF68 | 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. 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 8B/10B 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)

32.3.7.41 T_PCIE2_RP_PRIV_XP_LCTRL_2

Offset: 0xF6C | 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)

Bit	R/W	Reset	Description
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)
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)

Bit	R/W	Reset	Description
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)

32.3.7.42 T_PCIE2_RP_PRIV_XP_PAD_PWRUP

Offset: 0xF74 | 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

32.3.7.43 T_PCIE2_RP_PRIV_XP_RECOVERY_REASONS

This register records the reasons for going into the recovery state. Each bit represents a different reason. Search for "recovery_reason" in the linkfsm to see what each bit means, as it likely will change from project to project. Writing a 0 clears the register and enters into "record ALL recoveries encountered" mode. Writing a 1 clears the register and enters into "record NEXT recovery encountered" mode. The default mode is "record ALL".

Offset: 0xF84 | 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

32.3.7.44 T_PCIE2_RP_PRIV_XP_RECOVERY_COUNT

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: 0xF88 | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	0h	T_PCIE2_RP_PRIV_XP_RECOVERY_COUNT_VALUE: 0h: VALUE_INIT (default)

32.3.7.45 T_PCIE2_RP_PRIV_XP_RX_L0S_ENTRY_COUNT

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: 0xF8C | 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)

32.3.7.46 T_PCIE2_RP_PRIV_XP_TX_L0S_ENTRY_COUNT

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: 0xF90 | 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)

32.3.7.47 T_PCIE2_RP_PRIV_XP_L1_ENTRY_COUNT

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: 0xF94 | 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)

32.3.7.48 T_PCIE2_RP_PRIV_XP_L1_TO_RECOVERY_COUNT

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: 0xF98 | 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)

32.3.7.49 T_PCIE2_RP_PRIV_XP_L0_TO_RECOVERY_COUNT

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: 0xF9C | 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)

32.3.7.50 T_PCIE2_RP_PRIV_XP_L1P_ENTRY_COUNT

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: 0xFA0 | 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)

32.3.7.51 T_PCIE2_RP_PRIV_XP_ASLM_COUNT

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: 0xFA4 | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	0h	T_PCIE2_RP_PRIV_XP_ASLM_COUNT_VALUE: 0h: VALUE_INIT (default)

32.3.7.52 T_PCIE2_RP_RP_VEND_CTL2

This register contains miscellaneous control fields.

Offset: 0xFA8 | 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 maximum 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 its maximum 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)

Bit	R/W	Reset	Description
23	R/W	1h	T_PCIE2_RP_VEND_CTL2_IGNORE_ATTENTION_BUTTON_MSG: When this bit is set, the DUT will ignore upstream Attention button pressed message which has been sent by the endpoint. PCIe specification 1.1 specifies that this message should be ignored. 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 occurred which causes all further upstream transactions 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 transactions 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)
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

Bit	R/W	Reset	Description
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 maximum 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 merging is enabled on CPL packet (downstream). 1h: DIS_MA_TA_EP_MERGE_ON 0h: DIS_MA_TA_EP_MERGE_OFF (default)
0	R	0	Reserved

32.3.7.53 T_PCIE2_RP_PRIV_XP_CONFIG

Offset: 0xFAC | Read/Write: R/W

Bit	R/W	Reset	Description
31:2	R	0	Reserved
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. Primarily, this register supports the selection of different information for the low power duration count. It is used for selecting the Low power information to be presented in the T_PCIE2_RP_PRIV_XP_DURATION_IN_LOW_PWR_100NS register. 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

32.3.7.54 T_PCIE2_RP_PRIV_XP_DURATION_IN_LOW_PWR_100NS

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 register.

Note: This actually reports in multiples of clk25m time periods that fit into 100ns:

Actual time = int(100 / clk25m time period) * 100NS_VALUE

For example, when clk25m = 12 MHz, the time period is 83.33 ns, so this status register reports time = 100_NS_VALUE x 83.33 ns.

Offset: 0xFB0 | 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)

32.3.7.55 T_PCIE2_RP_PRIV_XP_SKP_TIMEOUT

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: 0xFB4 | 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

32.3.7.56 T_PCIE2_RP_PRIV_XP_EIDLE_INFERENCE_TIMEOUT

Offset: 0xFB8 | 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)

Bit	R/W	Reset	Description
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)

32.3.7.57 T_PCIE2_RP_VEND_SHADOW

This register is the shadow register of T_PCIE2_RP_SS_1. It must be programmed at boot time with the SubSystemVendorID and a unique non-zero number assigned by that vendor for this product. The T_PCIE2_RP_SS_1 register must be read-only at runtime in config space.

Offset: 0xFC8 | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R/W	0h	T_PCIE2_RP_VEND_SHADOW_SS_1_SSID: 0: SSID_INIT (default)
15:0	R/W	10DEh	T_PCIE2_RP_VEND_SHADOW_SS_1_SSVID: 10DEh: SSVID_INIT (default)

32.3.7.58 T_PCIE2_RP_TX_MARGIN_MAP0_TX_AMP

The MAP0 and MAP1 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: 0xFCC | 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
5:0	R/W	0h	T_PCIE2_RP_TX_MARGIN_MAP0_TX_AMP_CODE0: 0h: CODE0_INIT (default) 20h: CODE0_DESKTOP Ch: CODE0_MOBILE

32.3.7.59 T_PCIE2_RP_TX_MARGIN_MAP1_TX_AMP

Offset: 0xFD0 | 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

32.3.7.60 T_PCIE2_RP_ECTL_1_R1

Offset: 0xFD4 | Read/Write: R/W

Bit	R/W	Reset	Description
31:28	R/W	0h	T_PCIE2_RP_ ECTL_1_R1_TX_DRV_CNTL_1C: 0h: TX_DRV_CNTL_1C_DEFAULT (default)
20	R/W	0h	T_PCIE2_RP_ECTL_1_R1_RX_ QEYE_EN_1C: 0h: RX_ QEYE_EN_1C_DEFAULT (default)
18:16	R/W	0h	T_PCIE2_RP_ECTL_1_R1_RX_WANDER_1C: 0h: RX_WANDER_1C_DEFAULT (default)
15:14	R/W	0h	T_PCIE2_ RP_ECTL_1_R1_RX_TERM_CNTL_1C: 0h: RX_TERM_CNTL_1C_DEFAULT (default)
13:12	R/W	0h	T_PCIE2_RP_ECTL_1_R1_TX_TERM_CNTL_1C: 0h: TX_TERM_CNTL_1C_DEFAULT (default)
11:8	R/W	0h	T_PCIE2_RP_ECTL_1_R1_TX_CMADJ_1C: 0h: TX_CMADJ_1C_DEFAULT (default)
5:0	R/W	20h	T_PCIE2_RP_ECTL_1_R1_TX_AMP_1C: 20h: TX_AMP_1C_DEFAULT (default)

32.3.7.61 T_PCIE2_RP_ECTL_1_R2

Offset: 0xFD8 | Read/Write: R/W

Bit	R/W	Reset	Description
31:28	R/W	0h	T_PCIE2_RP_ ECTL_1_R2_TX_DRV_CNTL_1C: 0h: TX_DRV_CNTL_1C_DEFAULT (default)
20	R/W	0h	T_PCIE2_RP_ECTL_1_R2_RX_ QEYE_EN_1C: 0h: RX_ QEYE_EN_1C_DEFAULT (default)
18:16	R/W	0h	T_PCIE2_RP_ECTL_1_R2_RX_WANDER_1C: 0h: RX_WANDER_1C_DEFAULT (default)
15:14	R/W	0h	T_PCIE2_ RP_ECTL_1_R2_RX_TERM_CNTL_1C: 0h: RX_TERM_CNTL_1C_DEFAULT (default)
13:12	R/W	0h	T_PCIE2_RP_ECTL_1_R2_TX_TERM_CNTL_1C: 0h: TX_TERM_CNTL_1C_DEFAULT (default)
11:8	R/W	0h	T_PCIE2_RP_ECTL_1_R2_TX_CMADJ_1C: 0h: TX_CMADJ_1C_DEFAULT (default)
5:0	R/W	20h	T_PCIE2_RP_ECTL_1_R2_TX_AMP_1C: 20h: TX_AMP_1C_DEFAULT (default)

32.3.7.62 T_PCIE2_RP_TIMEOUT2

Offset: 0xFDC | 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 the minimum amount of time to stay in L1/L2. Holds off exit-condition detection for this amount of time to prevent accidental wake from data before E-Idle (second EIOS in Gen2) Measured in units of 4ns scaled with xclk frequency. Ah: MIN_L1_L2_IDLE_TIME_INIT (default)

32.3.7.63 T_PCIE2_RP_PRIV_MISC

Offset: 0xFE0 | 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 clock 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 propagate from the 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)

Bit	R/W	Reset	Description
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 (12MHz) 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. Drives the present status to the AFI: 1101 = No component present (Controller's internal post_x_bar_prsnt_ signal is hardwired to 1) 1110 = Component is present (Controller's internal post_x_bar_prsnt_ signal is hardwired to 0) 1111 = Default - post_x_bar_prsnt_ is no component present. (Controller's internal post_x_bar_prsnt_ signal is hardwired to 1) Fh: PRSNT_MAP_INIT (default)

32.3.7.64 T_PCIE2_RP_VEND_XP3

This register is the Symbol Alignment Error Handling register.

Offset: 0xFE8 | Read/Write: R/W

Bit	R/W	Reset	Description
31:8	R	0	Reserved
7:0	R/W	3h	T_PCIE2_RP_VEND_XP3_SA_ERROR_LIMIT: Specifies the number of misaligned COM symbols needed 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 8B/10B and symbol alignment errors are indistinguishable, symbol alignment errors (when error limit is hit) will contribute to all 8B/10B error counts including the Goto Recovery on 8B/10B errors feature. This is useful to speed up link retraining after losing symbol alignment. 3h: SA_ERROR_LIMIT_INIT (default)

32.3.7.65 T_PCIE2_RP_XP_CTL_1

This register contains control registers used in XP.

Offset: 0xFEC | 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, the 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)

Bit	R/W	Reset	Description
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 checking 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 the number of xclks for which the rx_data_en signals need to be deasserted before asserting 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 the link is operating in dynamically reduced link width and specification-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)
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 an 8B/10B 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 8B/10B 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 specification-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. 1h: ALLOW_SPEED_CHANGE_FROM_L0S_INIT (default)

Bit	R/W	Reset	Description
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 Root Port 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 field 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 field sets 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)
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)

32.3.7.66 T_PCIE2_RP_PRIV_XP_L1BEACON

Offset: 0xFF0 | 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 the 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

Bit	R/W	Reset	Description
0	R/W	0h	T_PCIE2_RP_PRIV_XP_L1BEACON_EN: 0h: EN_DEFAULT (default)

32.3.7.67 T_PCIE2_RP_TIMEOUT3

Offset: 0xFF4 | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R/W	Ch	T_PCIE2_RP_TIMEOUT3_RX_L0S_IDLE_TIME_GEN2: This field 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 field 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 field controls the minimum amount of time LTSSM stays in TX_L0S_IDLE state. Measured in units of clk25m clock periods (83 ns when using 12 MHz clock) 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 field 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)
3:0	R/W	0h	T_PCIE2_RP_TIMEOUT3_TX_EIDLE_EXIT_DELAY: This field 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)

32.3.7.68 T_PCIE2_RP_XP_CTL_2

This register contains controls bits used in XP.

Offset: 0xFF8 | 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. Forces the LTSSM training set detector to require PAD in the lane number symbol of training set in Config.LinkStart state: 0h: INIT (default)
7:0	R/W	40h	T_PCIE2_RP_XP_CTL_2_LOOPBACK_EXIT_THRESHOLD. Programs the maximum amount of time spent in the Loopback.Exit state as loopback slave. Units are 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. The default is 64 xclks: 40h: INIT (default)

32.3.7.69 T_PCIE2_RP_VEND_XP_STATS2

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: 0xFFC | 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 represents the corresponding logical lane. When set to 1, the 8B/10B 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

32.4 AFI Registers

32.4.1 AFI_AXI_BARi_SZ_0

There are 9 AXI BAR SIZE registers, where i = 0 through 8.

BAR0: Offset: 0x0 | Read/Write: R/W | Reset: 0x0003f000 (0b00111111000000000000)
BAR1 – BAR5: Offset: 0x4 + (i – 1) * 0x4, where i = 1 – 5 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000)
BAR6 – BAR8: Offset: 0x134 + (i – 6) * 0x4, where i = 6 – 8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000)

Bit	Description
19:0	AXI_BARi_SIZE: The size of the address range associated with BARi is in 4K increments. A value of 0 signifies BARi is not used.

32.4.2 AFI_AXI_BARi_START_0

There are 9 AXI BAR START registers, where i = 0 through 8.

BAR0: Offset: 0x18 | Read/Write: R/W | Reset: 0x01000000 (0b00000001000000000000)
BAR1 – BAR5: Offset: 0x1c + (i – 1) * 0x4, where i = 1 – 5 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000)
BAR6 – BAR8: Offset: 0x140 + (i – 6) * 0x4, where i = 6 – 8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000)

Bit	Description
31:12	AXI_BARi_START: The start of AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

32.4.3 AFI_FPCI_BARi_0

There are 9 FPCI BAR registers, where i = 0 through 8.

BAR0: Offset: 0x30 | Read/Write: R/W | Reset: 0x00010001 (0b00000000000000010000000000000xxx1)
BAR1 – BAR5: Offset: 0x34 + (i – 1) * 0x4, where i = 1 – 5 | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000000000000000xxx1)
BAR6 – BAR8: Offset: 0x14c + (i – 6) * 0x4, where i = 6 – 8 | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000000000000000xxx1)

Bit	Description
31:4	FPCI_BARi_START: The start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a shift left 8 of the value of this register.
0	FPCI_BARi_ACCESS_TYPE: Indicates if the address region is memory mapped versus configuration or I/O space. 1=Memory mapped access (PW only) 0=IO/config access (NPW only)

32.4.4 AFI_MSI_BAR_SZ_0

Do not use this reserved space.

MSI BAR Size

Offset: 0x60 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000)

Bit	Reset	Description
19:0	0x0	MSI_BAR_SIZE

32.4.5 AFI_MSI_FPCI_BAR_ST_0

MSI FPCI BAR Start

Offset: 0x64 | Read/Write: R/W | Reset: 0x58540000 (0b01011000010101000000000000000)

Bit	Reset	Description
31:4	0x5854000	MSI_FPCI_BAR_START: The start of upstream FPCI address space for MSI BAR. The upstream FPCI address is compared to start/1KB range for MSI BAR to determine if the access is MSI. Bits 31:4 of MSI BAR start correspond to UFPCI address bits 39:12.

32.4.6 AFI_MSI_AXI_BAR_ST_0

MSI AXI BAR Start

Offset: 0x68 | Read/Write: R/W | Reset: 0x80000000 (0b10000000000000000000)

Bit	Reset	Description
31:12	0x80000	MSI_AXI_BAR_START: The start of upstream AXI address space for MSI BAR. The upstream FPCI address is compared to start/1KB range for MSI BAR to determine if the access is MSI. Bits 31:12 of MSI BAR start correspond to AXI address bits 31:12.

32.4.7 AFI_MSI_VECi_0

MSI Vector

There are 8 MSI Vector registers (AFI_MSI_VECi), where i is a value from 0 to 7.

Offset: 0x6c + (i * 4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTORi: 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.

32.4.8 AFI_MSI_EN_VEC0_0

MSI Enable Vector

There are 8 MSI Enable Vector registers (AFI_MSI_EN_VECi), where i is a value from 0 to 7.

Offset: 0x8c + (i * 4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTORi: Each vector register corresponds to the enable bit for 32of 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.

32.4.9 AFI_CONFIGURATION_0

Configuration

Offset: 0xac | Read/Write: R/W | Reset: 0x000XXX40 (0b0xxxxxxxxxxx1xxx10xxxxxx01000000)

Bit	R/W	Reset	Description
31	RW	0x0	CLKEN_OVERRIDE: This can override the clock enable in case of malfunction.
19	RW	0x1	PW_NO_DEVSEL_ERR_CYA: Setting this bit disables detection of DECERR due to no devsel for DS PWs only.
18	RO	X	INITIATOR_READ_IDLE: This read-only bit provides status reads on AFI upstream. A value of 1b indicates there are no outstanding reads to the initiator.
17	RO	X	INITIATOR_WRITE_IDLE: This read-only bit provides status writes on AFI upstream. A value of 1b indicates there are no outstanding writes to the initiator.
15	RW	0x1	WDATA_LEAD_CYA: This bit is used to enable/disable the handling of write data ahead of requests on mselect
14	RW	0x0	WR_INTRLV_CYA: This bit is used to enable/disable the handling of interleaved write requests on mselect
13	RO	X	PE1_PRSNT_L_IN: This read-only bit is 0 when a card is present in PCIe slot 1
12	RO	X	PE0_PRSNT_L_IN: This read-only bit is 0 when a card is present in PCIe slot 0
11	RO	X	TARGET_READ_IDLE: This read-only bit provides status reads to AFI target. A value of 1b indicates there are no outstanding reads to the downstream FPCI.
10	RO	X	TARGET_WRITE_IDLE: This read-only bit provides status writes to AFI target. A value of 1b indicates there are no outstanding writes to the downstream FPCI.
9	RO	X	MSI_VEC_EMPTY: This read-only bit provides status on whether MSI Vector registers have any valid active bits.
7	RW	0x0	UFPCI_MSIAW: MSI After Write ordering rule. 1 = whenever MSI is ready assert the interrupt 0 = default behavior, apply MSIAW ordering rule
6	RW	0x1	UFPCI_PWPASSPW: Input to upstream FPCI 1 = whenever write is ready, send it; 0 = write goes only when outstanding PWs outside of new write's region are retired (default).
5	RW	0x0	UFPCI_PASSPW: Input to upstream FPCI. Allow upstream FPCI reads to pass writes.
4	RW	0x0	UFPCI_PWPASSNPW: Used for upstream FPCI. Allow upstream FPCI PWs to pass NPWs.
3	RW	0x0	DFPCI_PWPASSNPW: Used for downstream FPCI. Allow downstream FPCI PWs to pass NPWs.
2	RW	0x0	DFPCI_RSPPASSPW: Input to downstream FPCI. Allow downstream FPCI responses to pass writes
1	RW	0x0	DFPCI_PASSPW: Input to downstream FPCI. Allow downstream FPCI reads to pass writes.
0	RW	0x0	EN_FPCI: When the PCI device block is disabled, it is completely invisible on the PCI bus, i.e., it does not even process PCI configuration accesses.

32.4.10 AFI_FPCI_ERROR_MASKS_0

FPCI Error Masks

Offset: 0xb0 | Read/Write: R/W | Reset: 0x00000000 (0b000)

Bit	Reset	Description
2	0x0	MASK_FPCI_MASTER_ABORT: This bit allows an FPCI error to be forwarded to AXI response when the FPCI error response indicates Master Abort. 1 = forward error 0 = return AXI OKAY response (2'b0)
1	0x0	MASK_FPCI_DATA_ERROR: This bit allows an FPCI error to be forwarded to AXI response when the FPCI error response indicates Data Error. 1 = forward error 0 = return AXI OKAY response (2'b0)
0	0x0	MASK_FPCI_TARGET_ABORT: This bit allows an FPCI error to be forwarded to AXI response when

Bit	Reset	Description
		FPCI error response indicates Target Abort. This bit also covers a decode error generated when there is no devsel received. 1 = forward error, 0 = return AXI OKAY response (2'b0)

32.4.11 AFI_INTR_MASK_0

Interrupt Masks

Offset: 0xb4 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx0)

Bit	Reset	Description
8	0x0	MSI_MASK: MSI to MPCORE gated by mask.
0	0x0	INT_MASK: Interrupt to MPCORE gated by mask.

32.4.12 AFI_INTR_CODE_0

Interrupt Code

Detected errors are logged in an error signature register as specified here. The error signature register always stores the first error that occurred after enabling the error capture. The interrupt code register logs the interrupt ID shown in the ID column of Table 132. This register stores 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.

Offset: 0xb8 | Read/Write: R/W | Reset: 0x00000000 (0b00000)

Bit	Reset	Description
4:0	0x0	INT_CODE: If the code is 0, logging of the next interrupt is enabled. See Table 132 for the system messages. 0 = INT_CODE_CLEAR : Clear interrupt code 1 = INT_CODE_INI_SLVERR : Interrupt code for MPCORE AXI SLVERR response to AFI 2 = INT_CODE_INI_DECERR : Interrupt code for MPCORE AXI DECERR response to AFI 3 = INT_CODE_TGT_SLVERR : Interrupt code for PCIe endpoint FPCI target abort or data error response to AFI 4 = INT_CODE_TGT_DECERR : Interrupt code for PCIe2 FPCI master abort response to AFI 5 = INT_CODE_TGT_WRERR : Interrupt code for bufferable write to non-posted write address region 6 = INT_CODE_SM_MSG : Interrupt code for PCIe2 system management message 7 = INT_CODE_DFPCI_DECERR : Interrupt code for PCIe2 response to downstream request when the downstream FPCI address does not fall in a claimable downstream region 8 = INT_CODE_AXI_DECERR : Interrupt code for AFI response to downstream request when the mselect AXI address does not fall in any of AFI downstream BARs 9 = INT_CODE_FPCI_TIMEOUT : Interrupt code for FPCI Timeout 10 = INT_CODE_PE_PRSNT_SENSE : Interrupt code for Slot Present Pin Change 11 = INT_CODE_PE_CLKREQ_SENSE : Interrupt code for Slot Clock Request Change 12 = INT_CODE_CLKCLAMP_SENSE : Interrupt code for TMS Clock Clamp Change 13 = INT_CODE_RDY4PD_SENSE : Interrupt code for TMS Ready for power down 14 = INT_CODE_P2P_ERROR : Interrupt code for Peer2Peer errors

Table 132: System Messages

Error/Message	ID	Capture Information
AXI Slave error	1	Bits [31:2] of Initiator AXI address, direction of transaction
AXI Decode error	2	

Error/Message	ID	Capture Information
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
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

32.4.13 AFI_INTR_SIGNATURE_0

Interrupt Signature

Offset: 0xbc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000x0)

Bit	Reset	Description
31:2	0x0	INT_INFO: For interrupt codes 1-5/7-8, this field contains address bits [31:2], either in FPCI memory space or AXI space. If the interrupt code is 6, the information field INT_INFO[12:0] contains sideband information {sideband unitid, 3'b0, tms02sm_msg[4:0]}. For FPCI generated errors, the information field contains FPCI address. For AXI/AFI generated errors, the information field contains AXI address. For interrupt code 10, this field contains the status of slots 2-0 present pin. For interrupt code 11, this field contains the status of slots 2-0 clock request For interrupt code 12, this field contains level of clock clamp
0	0x0	DIR: Indicates the direction of the AXI/FPCI transaction. 1=rd 0=wr If signature type is 6 (sideband message), this field is 1. 0 = WRITE : Interrupt due to a write transaction 1 = READ : Interrupt due to a read transaction

32.4.14 AFI_UPPER_FPCI_ADDR_0

Upper FPCI Address

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxx00000000)

Bit	Reset	Description
17:16	0x0	P2P_ERR_RESP: These 2 bits are for the captured endpoint device error response (tms02ufa_cmd_error) when the interrupt code is 14.

Bit	Reset	Description
7:0	0x0	INT_INFO_UPPER: These 8 bits are the upper byte of captured FPCI address (bits[39:32]) when the interrupt code is 3, 4, or 7. These bits determine the region in the Hypertransport Address Map that was accessed.

32.4.15 AFI_SM_INTR_ENABLE_0

Sideband Message Interrupt Enable

Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

Bit	Reset	Description
14:0	0x0	ENABLE_MESSAGE: Each bit in this register enables a corresponding message shown in thetable 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		Uncorrectable Non-Fatal Error
	10		10		Uncorrectable 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

32.4.16 AFI_AFI_INTR_ENABLE_0

AFI Interrupt Enable

See the AFI_INTR_CODE_0 register for details on the interrupt codes.

Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000)

Bit	Reset	Description
12	0x0	EN_P2P_ERR: Enable bit for interrupt code 14
11	0x0	EN_RDY4PD_SENSE: Enable bit for interrupt code 13
10	0x0	EN_CLKCLAMP_SENSE: Enable bit for interrupt code 12
9	0x0	EN_PE_CLKREQ_SENSE: Enable bit for interrupt code 11
8	0x0	EN_PE_PRSNT_SENSE: Enable bit for interrupt code 10
7	0x0	EN_FPCI_TIMEOUT: Enable bit for interrupt code 9

Bit	Reset	Description
6	0x0	EN_AXI_DECERR: Enable bit for interrupt code 8
5	0x0	EN_DFPCI_DECERR: Enable bit for interrupt code 7
4	0x0	EN_TGT_WRERR: Enable bit for interrupt code 5
3	0x0	EN_TGT_DECERR: Enable bit for interrupt code 4
2	0x0	EN_TGT_SLVERR: Enable bit for interrupt code 3
1	0x0	EN_INI_DECERR: Enable bit for interrupt code 2
0	0x0	EN_INI_SLVERR: Enable bit for interrupt code 1

32.4.17 AFI_FPCI_TIMEOUT_0

Do not use this reserved space.

FPCI Timeout

Offset: 0xd8 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
31	0x0	SM2TMS0_FPCI_TIMEOUT_EN
19:0	0x0	SM2ALL_FPCI_TIMEOUT_THRESH

32.4.18 AFI_PCIE_THROTTLE_0

PCIe Throttle

Offset: 0xec | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx000000000000x000)

Bit	Reset	Description
31	0x0	SM2PCIE_THROT_EN: Override THERM MGMT
15:4	0x0	SM2PCIE_THROT_PERIOD: Override THERM MGMT period
2:0	0x0	SM2PCIE_THROT_DUTY_CYCLE: Override THERM MGMT duty cycle

32.4.19 AFI_PME_0

PCIe PME

Offset: 0xf0 | Read/Write: R/W | Reset: 0x00000XX0 (0bxxx0xxxxxxx0)

Bit	R/W	Reset	Description
11	RO	X	TMS0C12SM_PRESENCE_STATE: PCIe Link Presence State
10	RO	X	TMS0C12SM_PME_ACK: PCIe Endpoint PME Ack
9	RO	X	TMS0C12SM_PME: PCIe Endpoint PME message
8	RW	0x0	SM2TMS0C1_PME_TO: SM (system management) to PCIe PME Turn Off
6	RO	X	TMS0C02SM_PRESENCE_STATE: PCIe Link Presence State
5	RO	X	TMS0C02SM_PME_ACK: PCIe Endpoint PME Ack
4	RO	X	TMS0C02SM_PME: PCIe Endpoint PME message
0	RW	0x0	SM2TMS0C0_PME_TO: SM (system management) to PCIe PME Turn Off

32.4.20 AFI_REQ_PENDING_0

Request Pending

Offset: 0xf4 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxx)

Bit	Reset	Description
7	X	TMS0C12SM_NONISO_PENDING: SM (system management) status that a Non-ISO request is pending from PCIe to FPCI
6	X	TMS0C12SM_ISO_PENDING: SM (system management) status that an ISO request is pending from PCIE to FPCI
5	X	TMS0C12SM_NONCOH_REQUEST_PEND: SM (system management) status that a Non-coherent request is pending from PCIe to FPCI
4	X	TMS0C12SM_COH_REQUEST_PEND: SM (system management) status that a coherent request is pending from PCIe to FPCI
3	X	TMS0C02SM_NONISO_PENDING: SM (system management) status that a Non-ISO request is pending from PCIe to FPCI
2	X	TMS0C02SM_ISO_PENDING: SM (system management) status that an ISO request is pending from PCIe to FPCI
1	X	TMS0C02SM_NONCOH_REQUEST_PEND: SM (system management) status that a Non-coherent request is pending from PCIe to FPCI
0	X	TMS0C02SM_COH_REQUEST_PEND: SM (system management) status that a coherent request is pending from PCIe to FPCI

32.4.21 AFI_PCIE_CONFIG_0

PCIe Config

Offset: 0xf8 | Read/Write: R/W | Reset: 0x00003025 (0b0000xxx00011xxx00010x101)

Bit	Reset	Description
23:20	0x0	SM2TMS0_XBAR_CONFIG: SM (system management) configuration of PCIe crossbar. 1 = XBAR_4_1 : x4_x1 configuration 0 = XBAR_2_1 : x2_x1 configuration
16:12	0x3	UNITID_T0C1: T0C1 Upstream FPCI Unit ID. HyperTransport, upstream FPCI request. The downstream FPCI unit ID should remain 0.
8:4	0x2	UNITID_T0C0: T0C0 Upstream FPCI Unit ID. HyperTransport, upstream FPCI request
2	0x1	PCIEC1_DISABLE_DEVICE: Disables PCIe Controller 1 (default on)
1	0x0	PCIEC0_DISABLE_DEVICE: Disables PCIe Controller 0 (default off)
0	0x1	SM2TGIO_SLOT_EMPTY_PD_CYA: Indicates PCIe slot empty. Overrides PCIe slot present input.

32.4.22 AFI_REV_ID_0

Revision ID

Offset: 0xfc | Read/Write: R/W | Reset: 0x00000000 (0b00)

Bit	Reset	Description
1	0x0	CFG_REVID_WRITE_ENABLE: Write Enable for PCI backdoor rev ID override value.
0	0x0	CFG_REVID_OVERRIDE: Override for PCI config revision ID read-only register. This allows backdoor changes to rev ID for metal spins.

32.4.23 AFI_TOM_0

Top of Memory Limit

Offset: 0x100 | Read/Write: R/W | Reset: 0x3f3f0fff (0b111111001111111xxxx111111111111)

Bit	Reset	Description
29:16	0x3f3f	DLDT2ALL_TOM2: Top of Memory Limit 2. Determines peer-to-peer range as:{TOM2 :: 26'b0} to 0xFC_FFFF_FFFF
11:0	0xfff	DLDT2ALL_TOM1: Top of Memory Limit 1. Determines peer-to-peer range as:{TOM1 :: 26'b0} to 0xFFFF_FFFF (except MSI region)

32.4.24 AFI_FUSE_0

PCIe Fuse

Offset: 0x104 | Read/Write: R/W | Reset: 0x00000336 (0b011x011x11)

Bit	Reset	Description
10:8	0x3	FUSE_PCIE_WIDTH_T0C1: Configure PCIe as x1, x2, x4, x8, or x16. This should remain at 3'b011
6:4	0x3	FUSE_PCIE_WIDTH_T0C0: Configure PCIe as x1, x2, x4, x8, or x16. This should remain at 3'b011
2	0x1	FUSE_PCIE_T0_GEN2_DIS: Disable Gen 2 capability of PCIe.
1	0x1	FUSE_PCIE_SLI_DIS: Disable SLI capability for the GPU. This should remain on.

32.4.25 AFI_PMU_0

PMU Interface

Offset: 0x108 | Read/Write: R/W | Reset: 0x0XXXX000 (0bxxxxxxxxxxxxxx00000000xxx0)

Bit	R/W	Reset	Description
24	RO	X	CTLR_T0_C1_2PMU_TOG: PMU toggle response from PCIe
23:20	RO	X	CTLR_T0_C1_2PMU_STATUS: PMU Status
16	RO	X	CTLR_T0_C0_2PMU_TOG: PMU toggle response from PCIe
15:12	RO	X	CTLR_T0_C0_2PMU_STATUS: PMU Status
11:8	RW	0x0	PMU2CTLR_T0_C1_LOAD_INDICATOR_SCALE: PMU Load Indicator Scale for T0C1
7:4	RW	0x0	PMU2CTLR_T0_C0_LOAD_INDICATOR_SCALE: PMU Load Indicator Scale for T0C0
0	RW	0x0	PMU2ALL_LI_UPDATE_FAST_TOG: PMU Load Indicator Enable. This is used for wall-plug applications and should remain off.

32.4.26 AFI_PCIE_CLK_CONFIG_STATUS_0

PCIE2 CLK Config/Status

Offset: 0x10c | Read/Write: R/W | Reset: 0x0XXXXX00 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	R/W	Reset	Description
27:24	RO	X	PCIE2CLK_TMS0GRP2_PAD_MACRO_CLK_SEL: Clock select to pad macro. This should remain at 0.
23:20	RO	X	PCIE2CLK_TMS0GRP1_PAD_MACRO_CLK_SEL: Clock select to pad macro. This should remain at 0.
19:16	RO	X	PCIE2CLK_TMS0GRP0_PAD_MACRO_CLK_SEL: Clock select to pad macro. This should

Bit	R/W	Reset	Description
			remain at 0.
13	RO	X	PCIE2CLK_TMS0C1_DIS_XTXCLK1X: Request to gate T0C1 XTXCLK1X when in low power mode.
12	RO	X	PCIE2CLK_TMS0C0_DIS_XTXCLK1X: Request to gate T0C0 XTXCLK1X when in low power mode.
11	RO	X	PCIE2CLK_TMS0_CLAMP_CLK_L1: Request to gate TMS/FPCI clocks when in low power mode.
10	RO	X	PCIE2CLK_TMS0C1_SEL_XTXCLK1X_GEN2: Request to select Gen2 speed clock (500 MHz) for T0C1 XTXCLK1X. This is generated when register settings for PCIe2 specify Gen2 speed clocks.
9	RO	X	PCIE2CLK_TMS0C0_SEL_XTXCLK1X_GEN2: Request to select Gen2 speed clock (500 MHz) for T0C0 XTXCLK1X. This is generated when register settings for PCIe2 specify Gen2 speed clocks. This should remain low.
8	RO	X	PCIE2CLK_TMS0_SEL_XCLK_GEN2: Request to select Gen2 speed clock (500 MHz) for XCLK. This is generated when register settings for PCIe2 specify Gen2 speed clocks. This should remain low.
4	RW	0x0	CLK2PCIE_TMS0C1_OFF_XTXCLK1X: Acknowledge to disable T0C1 XTXCLK1X request. Used for clock gating.
3	RW	0x0	CLK2PCIE_TMS0C1_RDY_XTXCLK1X_GEN2: Acknowledge to select T0C1 XTXCLK1X Gen2 request. This should remain low.
2	RW	0x0	CLK2PCIE_TMS0C0_OFF_XTXCLK1X: Acknowledge to disable T0C0 XTXCLK1X request. Used for clock gating.
1	RW	0x0	CLK2PCIE_TMS0C0_RDY_XTXCLK1X_GEN2: Acknowledge to select T0C0 XTXCLK1X Gen2 request. This should remain low.
0	RW	0x0	CLK2PCIE_TMS0_RDY_XCLK_GEN2: Acknowledge to select XCLK Gen2 request. This should remain low.

32.4.27 AFI_PEX0_CTRL_0

PCIe PHY and Sideband Signal Interface

Offset: 0x110 | Read/Write: R/W | Reset: 0x00000000 (0b00x00)

Bit	Reset	Description
4	0x0	PEX0_REFCLK_OVERRIDE_EN: PEX0 enable to override refclk to be enabled always if PEX0_REFCLK_EN is set
3	0x0	PEX0_REFCLK_EN: PEX0 enable to clkout pad
1	0x0	PEX0_CLKREQ_EN: PEX0 enable clkreq to control clkout pad
0	0x0	PEX0_RST_L: PEX0 external pe0_rst_l register

32.4.28 AFI_PEX0_STATUS_0

Offset: 0x114 | Read/Write: RO | Reset: 0x0000000X (0bx)

Bit	Reset	Description
0	X	PEX0_CLKREQ_L: Status of the PEX0 pe0_clkreq_l input

32.4.29 AFI_PEX1_CTRL_0

Offset: 0x118 | Read/Write: R/W | Reset: 0x00000000 (0b00x00)

Bit	Reset	Description

Bit	Reset	Description
4	0x0	PEX1_REFCLK_OVERRIDE_EN: PEX1 enable to override refclk to be enabled always if PEX1_REFCLK_EN is set
3	0x0	PEX1_REFCLK_EN: PEX1 enable to clkout pad
1	0x0	PEX1_CLKREQ_EN: PEX1 enable clkreq to control clkout pad
0	0x0	PEX1_RST_L: PEX1 external pe1_rst_l register

32.4.30 AFI_PEX1_STATUS_0

Offset: 0x11c | Read/Write: RO | Reset: 0x0000000X (0bx)

Bit	Reset	Description
0	X	PEX1_CLKREQ_L: Status of the PEX1 pe1_clkreq_l input

32.4.31 AFI_INITIATOR_ISO_PW_RESP_PENDING_0

Initiator ISO PW Response Pending

Offset: 0x158 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:8	X	NUM_INITIATOR_ISO_PW_RESP_PEND_T0C1: Number of pending initiator ISO PW responses for controller 1
7:0	X	NUM_INITIATOR_ISO_PW_RESP_PEND_T0C0:Number of pending initiator ISO PW responses for controller 0

32.4.32 AFI_INITIATOR_NISO_PW_RESP_PENDING_0

Initiator Non-ISO PW Response Pending

Offset: 0x15c | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:8	X	NUM_INITIATOR_NISO_PW_RESP_PEND_T0C1: Number of pending initiator NISO PW responses for controller 1
7:0	X	NUM_INITIATOR_NISO_PW_RESP_PEND_T0C0: Number of pending initiator NISO PW responses for controller 0

32.4.33 AFI_PLLE_CONTROL_0

PLLE Controls

Offset: 0x160 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxx00)

Bit	Reset	Description
9	0x0	BYPASS_PADS2PLLE_CONTROL: Overrides PCIe PADS CLKREQ control of the PLLE.
8	0x0	BYPASS_PCIE2PLLE_CONTROL: Overrides PCIe2 CLOCK CLAMP control of the PLLE.
1	0x0	PADS2PLLE_CONTROL_EN: When active, all CLKREQs going inactive indicate DISABLE to the PLLE. When SATA, PCIe2, and PADS all indicate DISABLE to the PLLE, then the PLLE is disabled.
0	0x0	PCIE2PLLE_CONTROL_EN: When active, PCIE2 CLOCK CLAMP going active signal DISABLE to PLLE. When SATA, PCIe2, and PADS all indicate DISABLE to the PLLE, then the PLLE is disabled.

32.4.34 AFI_BUST_CONTROL_0

Bus Trace Module Controls

Offset: 0x164 | Read/Write: R/W | Reset: 0x00X00000 (0bx0xx00000000000000000)

Bit	R/W	Reset	Description
21	RO	X	PCIE_RXL_BUST_TRIG_OUT0_T0: Bus trigger from bus trace module in PCIe2
20	RW	0x0	TRACE_EXTERNAL_START: Start signal to bus trace module in PCIe2
17:16	RW	0x0	PCIE_RXL_BUST_BUS_TRACE_MUX_SEL_T0: MUX select to bus trace module in PCIe2
15:0	RW	0x0	CHIP_ID: Chip ID to bus trace module in PCIe2

32.4.35 AFI_PEXBIAS_CTRL_0

Offset: 0x168 | Read/Write: R/W | Reset: 0x00000001 (0b1)

Bit	Reset	Description
0	0x1	PEX_BIAS_PWRD: PEX clock bias pad power down

32.4.36 AFI_P2PBOM_CTRL_0

Offset: 0x16c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000)

Bit	Reset	Description
19:0	0x0	P2P_BASE:Peer-to-peer Bottom of Memory - connects to bits 39:20 of p2p_base_63_to_20 with the top bits tied to 0

32.4.37 AFI_P2PTOM_CTRL_0

Offset: 0x170 | Read/Write: R/W | Reset: 0x000003ff (0b00000000001111111111)

Bit	Reset	Description
19:0	0x3ff	P2P_LIMIT:Peer-to-peer Top of Memory - connects to bits 39:20 of p2p_limit_63_to_20 with the top bits tied to 0

32.4.38 AFI_CLKGATE_HYSTERESIS_0

Clock Gate Hysteresis

Offset: 0x174 | Read/Write: R/W | Reset: 0x00000014 (0b00010100)

Bit	Reset	Description
7:0	0x14	CLK_DISABLE_CNT: Number of AFI clock cycles to wait after clock gating criteria is met to disable the AFI/FPCI clocks

32.4.39 AFI_SPARE_REG0_0

Bit 31 is used as a Diagnostics bit to bypass both AXI target RAW and peer2peer RAW address comparison logic (default value is 1).

If 1, reads push all pending writes. Address range detection is bypassed.

If 0, reads only push pending writes that are in the same small address range.

Offset: 0x180 | Read/Write: R/W | Reset: 0xffff0000 (0b11111111111111110000000000000000)

Bit	Reset	Description
31:0	-65536	IPFS_SPARE_REG: Spare Register.

32.4.40 AFI_A2F_UFPCI_CFG0_0

Offset: 0x184 | Read/Write: R/W | Reset: 0x00000050 (0b00000000xxxxx0000000000001010000)

Bit	Reset	Description
31:24	0x0	STATIC_WAIT_IDLE_CNTR
18:16	0x0	STATIC_UFPCI_UFA_STARVE_CNTR_PRI1
15:12	0x0	STATIC_UFPCI_UFA_STARVE_CNTR_PRI0
11:10	0x0	STATIC_UFPCI_RR_BURST_SZ_PRI1
9:8	0x0	STATIC_UFPCI_RR_BURST_SZ_PRI0
7	0x0	STATIC_WAIT_CLAMP_EN
6	0x1	STATIC_UFPCI_UFA_DYN_BLOCK_EN
5	0x0	STATIC_UFPCI_UFA_BLK_COHERENT
4:2	0x4	STATIC_UFPCI_BLOCK_CMD_THRESHOLD
1	0x0	STATIC_CYA_UFA_ARB
0	0x0	STATIC_CYA_BACK2BACK_UPSTREAM_BLOCK

32.4.41 AFI_A2F_UFPCI_CFG1_0

Offset: 0x188 | Read/Write: R/W | Reset: 0x00000000 (0b00000000)

Bit	Reset	Description
7:0	0x0	STATIC_WAIT_UNCLAMP_CNTR

32.4.42 AFI_MSG_0

Reserved.

Offset: 0x190 | Read/Write: R/W | Reset: 0xXXX0XXX0 (0b000xxxxx0xxxxxxx000xxxxx0xxx)

Bit	R/W	Reset	Description
31	RW	0x0	TMS0C12SM_MSG_ERR_FATAL:PCIE FATAL ERROR MSG
30	RW	0x0	TMS0C12SM_MSG_ERR_NONFATAL:PCIE NONFATAL ERROR MSG
29	RW	0x0	TMS0C12SM_MSG_ERR_COR:PCIE CORRECTABLE ERROR MSG
28	RO	X	TMS0C12SM_MSG_RP_INT:PCIE ROOT PORT INTERRUPT
27	RO	X	TMS0C12SM_MSG_INTD:PCIE INTERRUPT D MSG
26	RO	X	TMS0C12SM_MSG_INTC:PCIE INTERRUPT C MSG
25	RO	X	TMS0C12SM_MSG_INTB:PCIE INTERRUPT B MSG
24	RO	X	TMS0C12SM_MSG_INTA:PCIE INTERRUPT A MSG
23	RW	0x0	TMS0C12SM_MSG_HOTPLUG_SCI:Pink HOTPLUG MSG
20	RO	X	TMS0C12SM_PM_PME:PCIE Endpoint PME message
15	RW	0x0	TMS0C02SM_MSG_ERR_FATAL:PCIE FATAL ERROR MSG
14	RW	0x0	TMS0C02SM_MSG_ERR_NONFATAL:PCIE NONFATAL ERROR MSG

Bit	R/W	Reset	Description
13	RW	0x0	TMS0C02SM_MSG_ERR_COR:PCIE CORRECTABLE ERROR MSG
12	RO	X	TMS0C02SM_MSG_RP_INT:PCIE ROOT PORT INTERRUPT
11	RO	X	TMS0C02SM_MSG_INTD:PCIE INTERRUPT D MSG
10	RO	X	TMS0C02SM_MSG_INTC:PCIE INTERRUPT C MSG
9	RO	X	TMS0C02SM_MSG_INTB:PCIE INTERRUPT B MSG
8	RO	X	TMS0C02SM_MSG_INTA:PCIE INTERRUPT A MSG
7	RW	0x0	TMS0C02SM_MSG_HOTPLUG_SCI:Pink HOTPLUG MSG
4	RO	X	TMS0C02SM_PM_PME:PCIE Endpoint PME message

32.4.43 AFI_AFI_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control (where applicable) and Clock Gating Control Register

Note: The FIFO timing aspects of this register are no longer supported, but retained for software compatibility

The clock override/ovr_mode fields of this register control the second-level clock gating for the client and MC side of the MCCIF. All clock gating is enabled by default.

A '1' written to the rclk/wclk override field will result in one of the following:

- With wclk/rclk override mode = LEGACY, the clock reverts to legacy mode of operation where the clock is on whenever the client clock is enabled.
- With wclk/rclk override mode = ON, the clock is always on inside MCCIF and PC. A '1' written to the cclk override field keeps client's clock always on inside MCCIF.

Offset: 0x7cc | Read/Write: R/W | Reset: 0x00000000 (0b00000xxxxxxxxxxxx0000)

Bit	Reset	Description
20	LEGACY	AFI_RCLK_OVR_MODE: 0 = LEGACY 1 = ON
19	LEGACY	AFI_WCLK_OVR_MODE: 0 = LEGACY 1 = ON
18	0x0	AFI_CCLK_OVERRIDE
17	0x0	AFI_RCLK_OVERRIDE
16	0x0	AFI_WCLK_OVERRIDE
3	DISABLE	AFI_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	AFI_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	AFI_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	AFI_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

32.4.44 AFI_ORDERING_RULES_0

This register is reserved.

Offset: 0x7d0 | Read/Write: R/W | Reset: 0x00000000 (0b000)

Bit	Reset	Description
3	0x0	UPSTREAM_MSIAW
2	0x0	UPSTREAM_RESPAW
1	0x0	UPSTREAM_RAW

[THIS PAGE INTENTIONALLY LEFT BLANK]

33.0 I2C CONTROLLER

The I^2C controller (I2C) implements an I^2C 3.0 specification-compliant I^2C master and slave controller. The I^2C controller supports multiple masters and slaves. It supports Standard mode (up to 100 Kbits/s), Fast mode (up to 400 Kbits/s), Fast mode plus (Fm+, up to 1 Mbits/s), and High-speed mode (up to 3.4 Mbits/s).

Tegra K1 devices have six instances of this controller. All six instances have identical I2C master functionality.

The I^2C controller supports DMA for the Master controller over the APB bus. There is no DMA support for the Slave controller. The I^2C controller also supports packet mode transfers where the data to be transferred is encapsulated in a predefined packet format as payload and sent to the I^2C controller over the APB bus. 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

- Standard (up to 100 Kbits/s), Fast (up to 400 Kbits/s), Fm+ (up to 1 Mbits/s), and High-speed (up to 3.4 Mbits/s) modes of operation.
- Clock stretching by the slave
- One to eight-word burst data transfers in DMA mode for the Master controller
- 4 Kbyte transfers in packet mode. Can be extended beyond 4 Kbytes by breaking up transfers into multiple packets (in both DMA and non-DMA modes)
- 7-bit or 10-bit addressing transactions
- Master is capable of data transfers to/from two or more slaves consecutively with a repeated-start condition
- Bus clear operations to address SDA issues
- General call addressing
- Recognition and transfer of data to peripherals that do not send an acknowledge (ACK)
- Master supports packet based DMA

Clocking

The I^2C controller has three clock domains:

- The Host interface runs on the APB clock (pclk). The APB clock is derived from the system clock and can be 1.0, ½, 1/3 or ¼ times the system clock (sclk).
- The I^2C interface controller clock runs on a frequency up to 136 MHz (maximum). The 136 MHz clock is derived from PLLP. Even though you can mux between PLLP, PLLC, PLLM, and OSC clocks, PLLP is always selected and used in normal operation.
- The time-out logic runs on a 32 kHz clock.

33.1 Functionality

The I^2C 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.

33.1.1 Bus Clear Master Operation

Bus clear logic helps to resolve I2C bus hang issues. If the data line (SDA) is permanently stuck low because a slave device is pulling it low continuously for some unknown reason, the I2C master loses the arbitration and stops driving the I2C bus. In this

case, software can use the Bus Clear master to get the SDA line released by sending clock pulses on the SCL line. Per the I2C specification, the device that held the bus Low should release it sometime within 9 clock pulses. But if the device needs more clock pulses than the default 9 pulses, software has an option to do so by programming a configurable register bit field (I2C_I2C_BUS_CLEAR_CONFIG_0[BC_SCLK_THRESHOLD]) to the required number of clock pulses.

Bus Clear Programming Procedure:

The Bus Clear operation starts with an ARB_LOST notification from the I^2C controller. The ARB_LOST notification could be due to many reasons. For example, in a multi-master platform, any other master could occupy the I2C bus already. When software initiates Tegra-I2C for a data transfer at this time, the master could backoff if it does not win the bus in the bus-arbitration procedure. Other cases for ARB_LOST are programming issues such as controller register configuration, pinmuxing, and pin tri-state clear.

If it was determined that ARB_LOST notification is due to a slave device continuously pulling the SDA line low, the bus clear operation can be initiated to recover the bus with the following steps:

1. Reset the I2C controller using module reset after the ARB_LOST notification from the controller.
2. Program the I2C_I2C_BUS_CLEAR_CONFIG_0[BC_SCLK_THRESHOLD] bit field to the required number of clock pulses.
3. Select the STOP or NO_STOP condition using the I2C_I2C_BUS_CLEAR_CONFIG_0[BC_STOP_COND] bit field.
4. Program I2C_I2C_BUS_CLEAR_CONFIG_0[BC_TERMINATE] as needed
5. Set the I2C_I2C_CONFIG_LOAD_0[MSTR_CONFIG_LOAD] register bit field to 1, and wait until this bit field is auto-cleared to zero by hardware.
6. Set the I2C_I2C_BUS_CLEAR_CONFIG_0[BC_ENABLE] bit field to start the bus clear operation. Hardware auto clears this bit after the bus clear operation is done.
7. Once enabled, the bus clear logic starts sending clock pulses on the SCL line.
8. Based on BC_TERMINATE settings, the Bus Clear logic:
   - IMMEDIATE: Sends clock pulses until SDA is released or the threshold count is reached, whichever is earlier or
   - THRESHOLD: Sends the threshold number of clock pulses irrespective of SDA line release status (released before threshold is reached or not-released with threshold count reached)
9. If SDA is released within the programmed number of clock pulses:
   - The Bus Clear logic terminates the transaction with STOP/NO-STOP condition based on the BC_STOP_COND settings.
   - Then it sets the bus_clear operation status in the I2C_I2C_BUS_CLEAR_STATUS_0[BC_STATUS] register bit field.
   - Finally, it sets the BUS_CLEAR DONE bit field in I2C_INTERRUPT_STATUS_REGISTER_0 and also interrupts the system if the corresponding interrupt enable bit is set in I2C_INTERRUPT_MASK_REGISTER_0. So software has to wait until it receives the BUS_CLEAR_DONE interrupt if the interrupt mechanism is used or the BUS_CLEAR_DONE status bit is set if the status-polling mechanism used before going for a BUS CLEAR operation successful/fail status check in I2C_I2C_BUS_CLEAR_STATUS_0[BC_STATUS].

33.1.2 Low-Power Operation

I^2C is a low-speed serial interface with only two bus lines (SCL, SDA) required. So from the I^2C specification side, there is no low-power operation specifically defined for the interface. But to save the power, use the lowest possible speed, if performance is not needed. The following bus speeds are supported by I^2C:

- Standard mode (Sm), with a bit rate up to 100 kbit/s
- Fast mode (Fm), with a bit rate up to 400 kbit/s
- Fast mode Plus (Fm+), with a bit rate up to 1 Mbit/s

- High-speed mode (Hs mode), with a bit rate up to 3.4 Mbit/s

33.1.2.1 Low Power Programming

Program the CLK_SOURCE_I2C register (for example: I2C1instance clock source address 0x60006124) and the I2C_CLK_DIVISOR registers such that the required data rate can be achieved at the minimum possible module clock so that power can be saved.

Below are the data relations for various modes of operation.

- Standard/Fast/Fm+:

  SCL = CLK_SOURCE.I2C / ((TLOW+THIGH+3)* (I2C_CLK_DIVISOR_STD_FAST_MODE+1)*I2C FREQUENCY DIVISOR) for lower values of I2C_CLK_DIVISOR_STD_FAST_MODE, up to 3

  SCL = CLK_SOURCE.I2C / ((TLOW+THIGH+2)* (I2C_CLK_DIVISOR_STD_FAST_MODE+1)*I2C FREQUENCY DIVISOR) for higher values of I2C_CLK_DIVISOR_STD_FAST_MODE, above 3

  Default values of I2C_I2C_INTERFACE_TIMING_0_0[TLOW] and I2C_I2C_INTERFACE_TIMING_0_0[THIGH] are 2 and 4 which would work for most of the data rates. So change them only when needed.
- HS mode:

  SCL = CLK_SOURCE.I2C / ((HS_TLOW+HS_THIGH+4) * (I2C_CLK_DIVISOR_HSMODE+1)*I2C FREQUENCY DIVISOR) for lower values of I2C_CLK_DIVISOR_HSMODE, up to 4

  SCL = CLK_SOURCE.I2C / ((HS_TLOW+HS_THIGH+2) * (I2C_CLK_DIVISOR_HSMODE+1)*I2C FREQUENCY DIVISOR) for higher values of I2C_CLK_DIVISOR_HSMODE, above 4The default values of I2C_I2C_HS_INTERFACE_TIMING_0_0[HS_TLOW] and I2C_I2C_HS_INTERFACE_TIMING_0_0[HS_THIGH] are 8 and 3 which would work for most of the data rates. So change them only when needed.

If there is a large rise time on the interface signals because of a weak pull-up or large load capacitance, then the SCL rate will be different from the above equations. The data rates will be slightly lower than expected.

33.1.2.2 IDLE Power Programming

When the module is not used or interface is idle, disable the module clock by clearing the CLK_ENB_I2C bits in the CAR module.

For example, the I2C1 clock enable bit is CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0[CLK_ENB_I2C1]. Software can clear this bit when it is seen that no transfers are needed from the I2C1 instance.

33.2 Software Interfaces

A typical peripheral controller might require the following steps to communicate or control an external peripheral:

1. Set up the registers
2. Program the ‘go’ bit to start the interaction with the external peripheral
3. The peripheral controller collects the response from the peripheral and interrupts software
4. Software reads the response

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 the hardware peripheral controller and software as described above can be classified into:

- ‘Request’ flow which can represent the register writes

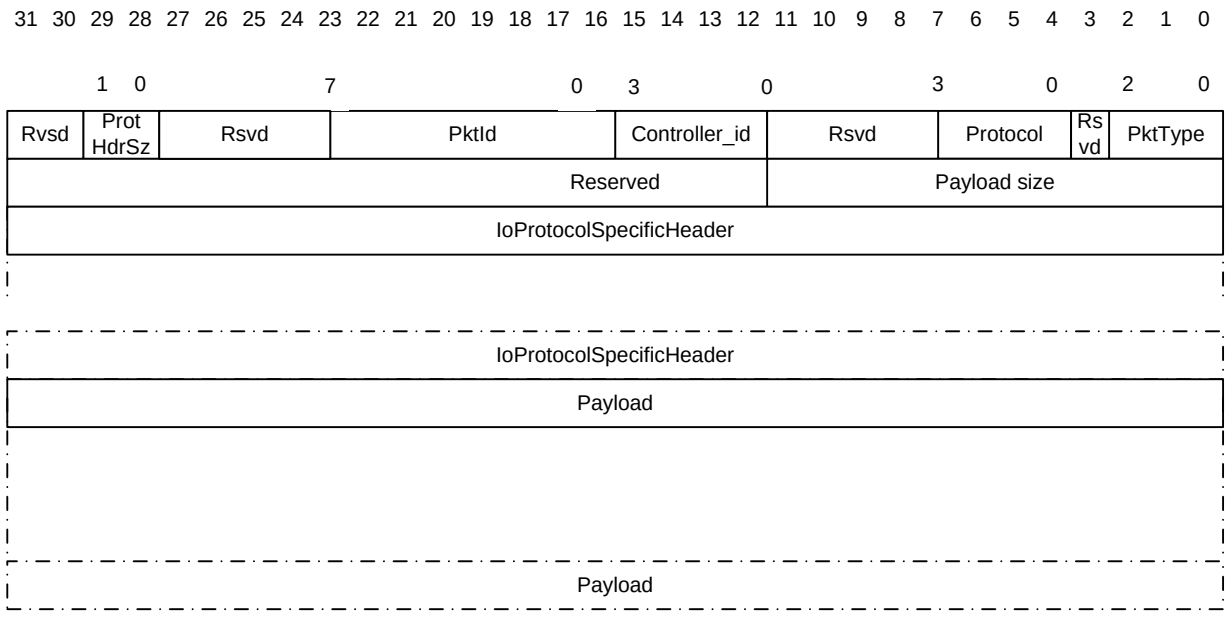
- ‘Response’ flow for peripheral controller responses sent back to software, and
- ‘Interrupt’ flow for out-of-band handshakes between hardware and software

Packets within a flow can be exchanged in a manner that requires full intervention from the CPU (PIO mode) or least intervention from the CPU (DMA mode). The I/O packet described below is a structure that can be used to represent various flows in the system.

33.2.2 I/O Packet Format

An I/O packet consists of a header and payload. The header consists of two parts: two words of generic header and 1-4 words of protocol-specific header.

Figure 120: 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 133: 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.

Bits	Field Value	Description
		The Controller ID field specifies the controller for which this packet is destined.
11:8	Reserved	Reserved
7:4	Protocol	The protocol is indicated in this field 0 = reserved 1= I2C 2-15 = reserved
3	Reserved	Reserved
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 Generic Header Word 1. The contents of Generic Header Word 1 are described in the table below.

Table 134: Generic I/O Header Word 1

Bits	Field Value	Description
31:12	Reserved	Reserved
11:0	PayloadSize	Payload size in bytes

Note about Payload Size:

For request packets that transmit data, the total packet size is payload_size + 8 bytes of generic header + protocol-specific header. However, for request packets with a read command, the payload size indicates the amount of data to be received. Hence the packet size = 8 bytes of generic header + protocol-specific header, because there is no payload in this packet, just the receive command.

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. The header is a minimum of three words with the first two words reserved for packet information and the third word reserved for protocol-specific requirements. A minimum of four words is needed to transfer one byte to the external peripheral.

33.2.4 Protocol-Specific Header

Depending on the implementation of the peripheral controller and the interface protocol, the header words can be defined. No restriction is placed on the organization of the fields or the number of protocol-specific words (a minimum of one word and a maximum of four words) that a particular implementation chooses.

I2C has one word with the Protocol-Specific Header. The contents of the I2C protocol-specific header are described in the next section.

33.2.5 I2C Master Packet Protocol Specific Headers

The I2C master supports two kinds of packets.

- Transmit packets that flow from Host to Controller.
- Receive data packets that flow from Controller to Host.

Table 135: I2C Master Transmit Packet Header

Bit	Name	Description
31:26	Reserved	Reserved
25	RESP_PKT_FREQ	Response packet frequency. This field is valid only if the I2C master is transmitting (READ/WRITE field below). If the I2C master is receiving, response packets are sent for every request packet received. 0: Send response packet at the end of last packet. 1: Send response packet for each packet transmitted.
24	RESP_PKT_ENABLE	Enable response packet. It qualifies RESP_PKT_FREQ field. 0: Do not send response packets (status has to be obtained via interrupt status bits). 1. Send response packet.
23	Reserved	Reserved
22	HS_MODE	Enable High speed mode (3.4 MHz operation)
21	CONTINUE_ON_NACK	Enable mode to handle devices that do not generate ACK upon the reception of a byte.
20	SEND_START_BYTE	1 = Send a start byte at the beginning of the transaction
19	READ/WRITE	1= READ
18	Address mode	1=10-bit mode 0 = 7-bit mode
17	IE	Generate interrupt upon packet completion
16	REPEAT_START/STOP	1 = Put a repeat start condition on the bus(to continue transaction) 0 = Put a stop condition on the bus
15	ContinueXfer	This bit overrides the REPEAT_START/STOP command above. 0: Introduce repeat start or stop condition based on bit 16. 1: Continue with current transfer without stop or repeat start condition.
14:12	HS_MASTER_ADDR	High Speed mode Master code
11:10	Reserved	Reserved
9:0	SLAVE_ADDR	Slave address. Bit 0 is ignored for 7-bit addressing, but should always match bit 19 (READ/WRITE).

Response packets contain the status of the I^2C controller. To accommodate that, the response packet header contains two words of the protocol-specific header.

Table 136: I2C Master Response Packet Header Word 0

Bit	Name	Description
31:27	Reserved	Reserved
26	RFIFO_OVF	Receive FIFO overflowed.
25	TFIFO_OVF	Transmit FIFO overflowed.
24	TRANSFER_COMPLETE	All the packets have been transferred successfully. The packet for which the “LAST PACKET” field is set has been transferred.
23:16	TRANSFER_PKT_ID	The current packet ID for which the transaction is happening on the bus
15:4	TRANSFER_BYTENUM	The number of bytes transferred in the current packet
3	NOACK_FOR_ADDR	1 = No ACK received for the address byte
2	NOACK_FOR_DATA	1 = No ACK received for the current data byte
1	ARB_LOST	1 = Arbitration lost
0	CONTROLLER_BUSY	1 = Controller is busy

33.3 Programming Guidelines

33.3.1 I2C Frequency Divisor Register

The FREQUENCY DIVISOR register (CLK_SOURCE_I2C register, whose address is: 0x60006124 for I2C1) must be programmed as a function of the CLK_SOURCE selected for I2C as follows:

```
Standard/Fast-mode/Fm+:
SCL = CLK_SOURCE.I2C / ((TLOW+THIGH+3)* (I2C_CLK_DIVISOR_STD_FAST_MODE+1)*I2C FREQUENCY DIVISOR) for
lower values of I2C_CLK_DIVISOR_STD_FAST_MODE, up to 3
SCL = CLK_SOURCE.I2C / ((TLOW+THIGH+2)* (I2C_CLK_DIVISOR_STD_FAST_MODE+1)*I2C FREQUENCY DIVISOR) for
higher values of I2C_CLK_DIVISOR_STD_FAST_MODE, above 3
Default values of I2C_I2C_INTERFACE_TIMING_0_0[TLOW] and I2C_I2C_INTERFACE_TIMING_0_0[THIGH] are 2 and 4
which works for most of the data rates. Change them only when needed.
High Speed mode:
SCL = CLK_SOURCE.I2C / ((HS_TLOW+HS_THIGH+4) * (I2C_CLK_DIVISOR_HSMODE+1)*I2C FREQUENCY DIVISOR) for
lower values of I2C_CLK_DIVISOR_HSMODE, up to 4
SCL = CLK_SOURCE.I2C / ((HS_TLOW+HS_THIGH+2) * (I2C_CLK_DIVISOR_HSMODE+1)*I2C FREQUENCY DIVISOR) for
higher values of I2C_CLK_DIVISOR_HSMODE, above 4
Default values of I2C_I2C_HS_INTERFACE_TIMING_0_0[HS_TLOW] and I2C_I2C_HS_INTERFACE_TIMING_0_0[HS_THIGH]
are 8 and 3 which works for most of the data rates. Change them only when needed.
If there is a large rise time on the interface signals because of a weak pull-up or large load
capacitance, the SCL rate would be different from the above equations. You would notice slightly lower
data rates than expected.
```

The clock enable (bit 12 of the CLK_OUT_ENB.L register) must also be given to the I^2C controller, before any of the registers are written.

Table 137: Recommended Settings for Various I2C Speeds

Source	PLLP_OUT0	I2C Source Div	Tlow+Thigh	SM/FM Div	HS Div	I2C Frequency	Debounce
PLLP	408MHz	3 (0x02)	12[1]	N/A	3 (0x02)	3.48MHz (HS)	0
PLLP	408MHz	3 (0x02)	8	17 (0x10)	N/A	1MHZ (FM+)	0
PLLP	408MHz	5 (0x04)	8	26 (0x19)	N/A	392KHz (FM)	2
PLLP	408MHz	20 (0x13)	8	26 (0x19)	N/A	98KHz (SM)	2

33. The HS_THIGH field in the I2C_I2C_HS_INTERFACE_TIMING_0_0 register must be set to 2 (from default 3) to achieve the full 3.48MHz speed in HS mode.

Debounce will impact the I2C interface data rate in the following way.

$$I2C\ SCL\ rate = \frac{CLK\_SOURCE}{SOURCE\_DIV * (Tlow + Thigh + 2 + Round\_Trip\_Delay) * STD\_or\_HS\_DIV}$$

Where:

$$Round\_Trip\_Delay\ in\ I2C\ Div\ \text{cycles} = \frac{(Chip\_internal\_delay + Load\_Delay + (((Debounce\_Cnt * 2 + 1) + Synchronizer) * I2C\_Clk\_Period))}{I2C\_Clk\_Period}$$

For example, if CLK_SOURCE = 408 MHz and Source Div = 5:

I2C_Clock frequency = 408/5 = 81.6 MHz ➔ I2C clock period = 12.25 ns

Chip internal delay = 10 ns, Load_Delay = 55 ns, Debounce = 2, Synchronizer Delay = 3

Round_Trip_Delay = (10 ns + 55 ns + 8*12.25 ns) = ~163 ns

FM+ speed case: STD_FM_Div = 10

STD_FM_Div * I2C_Clk_Period = 10*12.255 ns = 122.55 ns

Round_Trip_Delay in I2C_Div cycles = Round_Trip_Delay/(STD_FM_Div * I2C_Clk_Period) = 1.

So I2C SCL rate = $\frac{408MHz}{5*(8+1)*10}$ = 906 KHz

If Debounce is disabled, the Round_Trip_delay (in I2C_Div cycles) becomes zero.

So I2C SCL rate = $\frac{408MHz}{5*(8)*10}$ = 1002 KHz = 1.02 MHz

Table 138 I2C Data Rate Variation with Debounce

I2C Mode	Clock Source	Source Clock Frequency	I2C Source Divisor	SM/FM Divisor	Debounce Value	I2C SCL Frequency
FM+	PLLP_OUT0	408MHz	5 (0x04)	10 (0x9)	0	1016KHz
					[5:1]	906KHz
					[7:6]	816KHz
FM	PLLP_OUT0	408MHz	5 (0x4)	26 (0x19)	[7:0]	392KHz
SM	PLLP_OUT0	408MHz	20 (0x13)	26 (0x19)	[7:0]	98KHz
				HS Divisor		
HS	PLLP_OUT0	408MHz	3 (0x2)	3 (0x2)	0	3.48MHz
					[7:1]	Not allowed

33.3.2 I2C Command ADDR Registers

These registers, I2C_CMD_ADDR0 and I2C_CMD_ADDR1, contain the address of the slave with which a transaction is intended. The I^2C Master Controller can be programmed to transact with both 7-bit and 10-bit addressed slaves. Bit [0] of the I2C_CNFG register determines the size of the slave address to be transacted. If a 7-bit or a 10-bit slave address is chosen, the respective address is taken from I2C_CMD_ADDR0 [9:0], and the Read/Write command is taken from bits 6 and 7 of the I2C_CNFG Register. In a 2 Slave configuration, the slave 1 address (7-bit or 10-bit) is taken from I2C_CMD_ADDR0 [9:0] and Slave 2 address (7-bit or 10-bit) from I2C_CMD_ADDR1 [9:0]. The transfer length is the same in 2 slave operation and is taken from the I2C_CNFG [LENGTH] bit field. In 2-slave operation, the maximum possible transfer size is 4 bytes.

33.3.3 I2C Data Registers

There are two data registers, I2C_CMD_DATA1and I2C_CMD_DATA2, each with 32-bit length, which are to be loaded with the data to be transmitted or received (I2C_SL_RCVD register).

33.3.4 I2C Configuration Register

This register is to be configured with the following data:

- Number of bytes to be transmitted or received
- Select for 7-bit or 10-bit slave address
- Program to send a Start Byte
- Single or two slave operations
- Support of NOACK from an external peripheral

The I2C_CNGF [9] bit is used to issue a “Begin-Transaction” command to the I^2C Master Controller. This bit is auto cleared by the hardware, and other bits of the register are masked for writes, when this bit is programmed to be one. Hence, the firmware should first configure all the other registers and bits [8:0] of the I2C_CNFG register before the I2C_CNFG [9] bit is programmed to one.

33.3.5 I2C Configuration Load Register

This CONFIG load register transfers the software programmed configuration in the I2C registers to hardware internal registers used in the logic. It has three bit fields:

- MSTR_CONFIG_LOAD for regular master and Bus Clear master logic
- SLV_CONFIG_LOAD for slave controller logic
- TIMEOUT_CONFIG_LOAD for SMBUS timeout logic.

Software must set these fields to 1 for the actual register configuration to take effect. Thus software is programming only shadow registers through regular configuration. When these load_config bit fields are set to 1, it causes the regular/shadow registers configuration to be transferred to the hardware internal active registers. So software has to set these bit fields at the end of all regular register configurations. However, in normal or non-packet mode, these bit fields should be set to 1 before the I2C_I2C_CNFG_0[SEND] bit is set to 1. The I2C_I2C_CNFG_0[SEND] bit triggers a transfer on the I2C bus and should be programmed at the end of the entire configuration. These config_load bits are hardware auto-clear bits. Hardware clears these bit fields once the register configuration is moved to hardware internal active registers. So software must wait until these bits are auto-cleared before continuing with further programming.

Below are the register bit fields associated with each CONFIG_LOAD bit.

MSTR_CONFIG_LOAD needs to be set for the following registers:

- I2C_I2C_CNFG_0
- I2C_I2C_CMD_ADDR0_0
- I2C_I2C_CMD_ADDR1_0
- I2C_I2C_CMD_DATA1_0
- I2C_I2C_CMD_DATA2_0
- I2C_I2C_CLK_DIVISOR_REGISTER_0
- I2C_I2C_BUS_CLEAR_CONFIG_0
- I2C_I2C_INTERFACE_TIMING_0_0
- I2C_I2C_INTERFACE_TIMING_1_0
- I2C_I2C_HS_INTERFACE_TIMING_0_0
- I2C_I2C_HS_INTERFACE_TIMING_1_0

SLV_CONFIG_LOAD covers the following registers:

- I2C_I2C_SL_CNFG_0
- I2C_I2C_SL_ADDR1_0
- I2C_I2C_SL_ADDR2_0
- I2C_I2C_TLOW_SEXT_0[RST_SL_ON_TIMEOUT]
- I2C_I2C_SL_DELAY_COUNT_0

TIMEOUT_CONFIG_LOAD covers the following register bit fields:

- I2C_I2C_TLOW_SEXT_0[TIMEOUT]
- I2C_I2C_TLOW_SEXT_0[TLOW_SEXT]
- I2C_I2C_TLOW_SEXT_0[TLOW_MEXT]

- I2C_I2C_TLOW_SEXT_0[TIMEOUT_EN]
- I2C_I2C_TLOW_SEXT_0[TLOW_SEXT_EN]
- I2C_I2C_TLOW_SEXT_0[TLOW_MEXT_EN]

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 controller and the completion status of the command executed.

33.3.7 I2C Interface Timing Registers

These registers provide flexibility to tune the I2C interface bus timing, if needed. Separate bit fields are provided for non-HS (STD/FM/FM+) and HS modes timing.

33.3.8 Interrupt Generation in Non-Packet (Normal) Mode

In non-packet mode, 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 the transaction, as programmed in the LENGTH field of the I2C_CNFG register, is complete.

33.3.8.1 Example Transfers in Normal Mode

Write Example:

This programming example is for 7 byte writes from master to an external slave.

Program the I2C_CLK_DIVISOR register to get the required data rate based on I2C CLK_SOURCE register programming in CAR module

- Write the slave address in the I2C_CMD_ADDR0 register based on 7-bit/10-bit addressing mode
- Write the first 4 bytes of data in the I2C_CMD_DATA1 register
- Write the remaining 3 bytes in I2C_CMD_DATA2 register
- Program I2C_CNFG[DEBOUNCE_CNT] to the required value as needed
- Set I2C_CNFG[A_MOD] to 7-bit or 10-bit addressing
- Program I2C_CNFG [LENGTH]. This bit field works in (n+1) fashion; so for a 7-byte transfer, 6 needs to be programmed in the LENGTH bit field
- Set I2C_CNFG [SLV2] = 0 as this is one slave access
- Set CMD1 = 0 for write operation
- Set NOACK to 0 or 1 based on Slave type
- Set the MSTR_CONFIG_LOAD bit in the I2C_CONFIG_LOAD register
- Finally, set I2C_CNFG [SEND] to 1 to begin a write transaction on the interface
- Wait unit the transaction is complete, either until an interrupt is received or the I2C_STATUS[BUSY] bit becomes zero.
- Check I2C_STATUS [CMD1_STAT] to see if the transaction is successful or there is a NOACK from the slave for any of the bytes transferred.

Read Example

A read operation is divided into two steps based on random read or immediate read after a write. In a typical random read:

1. Write command : Send slave register/memory index first from which the data needs to be read

2. Send read command to read the data. This operation can be done in two separate configurations or in a single configuration using Repeated Start in 2-slave config.

These steps are described in more detail below.

- Write: Sending slave's 1-byte register index first
  - Write the slave address in I2C_CMD_ADDR0 register based on 7-bit/10-bit addressing mode
  - Write the slave's internal register index (from which the data has to be read) in the I2C_CMD_DATA1 register
  - Program I2C_CNFG[DEBOUNCE_CNT] to the required value as needed
  - Set I2C_CNFG[A_MOD] to 7-bit or 10-bit addressing
  - Program I2C_CNFG[LENGTH] = 0 for a 1 byte register index transfer
  - Set I2C_CNFG[SLV2] = 0
  - Set I2C_CNFG[CMD1] = 0 for write operation
  - Set I2C_CNFG[NOACK] to 0 or 1 based on the Slave type
  - Finally, set I2C_CNFG[SEND] to begin the write transaction on the interface
  - Wait unit the transaction is complete (wait until either an interrupt is received or the I2C_STATUS [BUSY] bit becomes zero).
  - Check I2C_STATUS[CMD1_STAT] to see if the transaction is successful or there is a NOACK from the slave
- Read: Send read command
  - Program I2C_CNFG[LENGTH] = 6 for 7-bytes read
  - Set I2C_CNFG[SLV2] = 0 because this is one slave access
  - Set I2C_CNFG[CMD1] = 1 for read operation
  - Finally set I2C_CNFG[SEND] to begin write transaction on the interface
  - Wait unit the transaction is complete (wait until either an interrupt is received or the I2C_STATUS [BUSY] bit becomes zero).
  - Check I2C_STATUS[CMD1_STAT] to see if the transaction is successful or there is a NOACK from the slave
  - Read the first 4 bytes data in the I2C_CMD_DATA1 register
  - Read the remaining 3 bytes in the I2C_CMD_DATA2 register
- Read using repeated-start in 2-slave mode
  - Write the slave address in the I2C_CMD_ADDR0 register with the LSB bit set to "0"
  - Write the slave address in the I2C_CMD_ADDR1 register with the LSB bit set to "1"
  - Write the slave's internal register index byte (from which the data has to be read) in the I2C_CMD_DATA1 register
  - Program I2C_CNFG[DEBOUNCE_CNT] to the required value as needed
  - Set I2C_CNFG[A_MOD] to 7-bit or 10-bit addressing
  - Program I2C_CNFG[LENGTH] = 0 for 1 byte transfer
  - Set I2C_CNFG[SLV2] = 1
  - Set I2C_CNFG[CMD1] = 0 for write operation
  - Set I2C_CNFG[CMD2] = 1 for read operation
  - Set I2C_CNFG[NOACK] to 0 or 1 based on the Slave type
  - Finally set I2C_CNFG[SEND] to begin the write transaction on the interface
  - Wait unit the transaction is complete (wait until either an interrupt is received or the I2C_STATUS [BUSY] bit becomes zero).

- Check I2C_STATUS [CMD1_STAT] to see if the transaction is successful or there is a NOACK from the slave
- Read data from the I2C_CMD_DATA2 register

33.4 Programming Guidelines for Packet-Based Interface

33.4.1 Packet Based Interface Registers

The following registers are used for a packet based interface:

- I2C TX FIFO
- I2C RX FIFO
- PACKET TRANSFER STATUS
- FIFO CONTROL
- FIFO STATUS
- FIFO STATUS (common for both master and slave)
- Interrupt Mask Register (common for both master and slave)
- Interrupt Status Register (common for both master and slave)
- I2C Clock Divisor Register for master
- PACKET_MODE_EN, DEBOUNCE_CNT, and NEW_MASTER_FSM fields of I2C_CNFG Register
- CONFIG_LOAD bits in the I2C_CONFIG_LOAD 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

Software should program the I2C_TX_PACKET_FIFO register. It writes the packet header followed by the payload. The header size can vary from 3 to 5 words. For I2C, the size is 3 words for request packets and 4 words for response packets. The first two words of the header contain a generic header. The third word of the packet (and fourth as well for response packets) contains I2C transaction-specific information. The payload contains the actual data to be written to the slave. In case of read operations, payload is nil, and hence, the packet contains only a header.

33.4.3 Reading from the RX FIFO

The data received on the bus is pushed into the RX FIFO. In PIO mode, a request interrupt is generated when the attention level is reached, and software reads the RX_FIFO register to get the data. In DMA mode, a request is asserted to DMA after the attention level is reached.

33.4.4 Error Handling

The following steps describe what needs to happen when an error occurs due to a NOACK, loss of arbitration, TX FIFO overflow, or RX FIFO overflow before proceeding with any further transactions:

- When an error occurred due to a NOACK, a stop condition is put on the bus and an interrupt is generated
- The status register is updated with the current error packet ID at which the error occurred
- Software should reset the controller
- In case of a DMA transfer, DMA needs to be restarted

In packet mode, special care should be taken to handle error recovery. If an error occurs during the transfer of a packet, then the controller should be reset, DMA needs to be restarted (if using DMA transfers), and the entire packet needs to be resent since there is no accurate way of knowing how many bytes made it to the I2C slave.

The repeat start bit in packet mode can complicate the situation further. When back-to-back packets are sent to a slave with the repeat start bit set, and if an error occurs, it is not always known during which packet transfer the error occurred. Therefore, the entire stream of packets that were sent back to back using the repeat start bit need to be sent again.

Receives are simpler, since if an error occurs, and bytes for a packet have already been received, that packet can be deemed complete. Only incomplete transfers need to be retried.

33.4.5 Programming REPEAT_START/STOP

This field indicates whether or not to put a stop or repeated-start condition after the current transaction. By default, this bit is zero and a stop condition is put on the bus. If this bit is set, a repeat start is put on the bus before proceeding with the next packet. The REPEAT_START/STOP bit can be used for combining read operations and write operations within a single transaction with a repeat start, or to do transfers beyond the 4 Kbyte limit. This bit is present in the protocol-specific header.

33.4.6 Programming Continue-Xfer

The I^2C controller supports transfer sizes beyond 4 Kbytes without a repeat start condition, again by combining multiple packets. This is in addition to the ability to use a repeat start condition to do continuous transfers over 4 Kbyte limit. With the ContinueXfer field, the current transfer can continue without a stop or a repeat start condition.

33.4.7 FIFO Control

In DMA mode, TX_FIFO_TRIG indicates the number of words that need to be empty in the TX FIFO for the DMA trigger to be asserted. RX_FIFO_TRIG indicates the number of words that need to be full in the RX FIFO for the DMA trigger to be asserted.

In PIO mode, TFIFO_DATA_REQ in the Interrupt Status Register will be set if the TX FIFO empty count is more than the value programmed in TX_FIFO_TRIG. Similarly RFIFO_DATA_REQ in the Interrupt Status Register will be set if the RX FIFO full count is more than the value programmed in RX_FIFO_TRIG. The CPU will be interrupted if the status bits are set and their corresponding INT_EN bit is set in the Interrupt Mask Register. The depth of the TX and RX FIFOs is 8.

33.4.8 FIFO Status

This register indicates the number of 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).

33.4.9 Example Transfer

Here is a programming example for a 1 byte read followed by a 5-byte write

1. Program the PACKET_MODE_EN field to 1 and then set the corresponding LOAD_CONFIG bit fields to 1.
2. Write the generic packet header
3. Write the payload size = 0x0 (1 byte)
4. Write the I2C specific header read/write = 1 and repeat_start/stop = 1
5. Write the generic packet header
6. Write payload size = 0x4 (5 bytes)
7. Write the I2C specific header read/write = 0 and repeat_start/stop = 0
8. Write the 5 bytes (word aligned)

33.5 I2C Registers

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

33.5.1 I2C_I2C_CNFG_0

IC Controller Configuration Register (Master)

The I2C_CNFG register is used to configure:

- the number of bytes to be transmitted or received
- the slave device type (either a 7-bit device or a 10-bit device)
- enable mode to send Start-byte or not
- to select either a single slave transaction or two slave transaction
- enable mode to handle devices that do not generate an ACK.

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000800 (0bxxxxxxxxxxxxxxxx0000100000000000)

Bit	Reset	Description
15	0x0	MSTR_CLR_BUS_ON_TIMEOUT: When this bit is set, the I2C master will force the clock low for an extended period of time (>TIMEOUT) to force all SMB slaves to release the bus. 0 = Do not clear the bus on TLow: SEXT/TIMEOUT time-out 1 = Clear the bus on TLow: SEXT/TIMEOUT time-out
14:12	0x0	DEBOUNCE_CNT: Debounce period for SDA and SCL lines. 0 = No debounce 1 = 2T 2 = 4T 3 = 6T, etc. where T is the period of the fixed PLL clock source coming to I2C. Maximum debounce period is 14T. A debounce period of > 50 ns is desirable.
11	0x1	NEW_MASTER_FSM: Maintained for backward compatibility. 0 = DISABLE 1 = ENABLE
10	0x0	PACKET_MODE_EN: Write 1 to initiate transfer in packet mode. 0 = NOP 1 = GO
9	0x0	SEND: Writing a 1 causes the master to initiate the transaction in normal mode. This bit is cleared by hardware when the transaction is done. Other bits of the register are masked for writes when this bit is programmed to one. Thus, firmware should first configure all other registers and bits [8:0] of this register before this bit is programmed to one. 0 = NOP 1 = GO
8	0x0	NOACK: Enable mode to handle devices that do not generate an ACK. 1 = Do not look for an ACK at the end of the enable. 0 = DISABLE 1 = ENABLE
7	0x0	CMD2: Read/Write Command for Slave 2: 1 - Read Transaction; 0 - Write Transaction. For a 7-bit slave address, this bit must match with the LSB of address byte for slave 2. Valid only when bit 4 of this register is set.
6	0x0	CMD1: Read/Write Command for Slave 1: 1 - Read Transaction; 0 - Write Transaction. Command for Slave 1: For a 7-bit slave address, this bit must match with the LSB of address byte for slave 1.
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.

Bit	Reset	Description
3:1	0x0	LENGTH: The number of bytes to be transmitted per transaction. 000 = 1 byte ... 111 = 8 bytes. In a two slave transaction, the number of bytes should be programmed to be less than 011.
0	0x0	A_MOD: Address mode defines whether a 7-bit or a 10-bit slave address is programmed. 1 = 10-bit device address. 0 = 7-bit device address.

33.5.2 I2C_I2C_CMD_ADDR0_0

I2C Slave-1 Address

I2C_CMD_ADDR0 is programmed with the 7-bit or 10-bit address of slave 1 to which the transaction is intended.

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:0	0x0	ADDR0: For 7-bit mode, the address is written in I2C_CMD_ADDR0[7:1], and I2C_CMD_ADDR0[0] indicates the read/write transaction. The I2C_CMD_ADDR0[0] bit must match the I2C_CNFG[6] bit. For 10-bit mode, the address is written in I2C_CMD_ADDR0[9:0], and I2C_CNFG[6] indicates the read/write transaction.

33.5.3 I2C_I2C_CMD_ADDR1_0

I2C Slave-2 Address

I2C_CMD_ADDR1 is programmed with the 7-bit or 10-bit address of slave 2 to which the transaction is intended.

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:0	0x0	ADDR1: For 7-bit mode, the address is written in I2C_CMD_ADDR0[7:1], and I2C_CMD_ADDR0[0] indicates the read/write transaction. The I2C_CMD_ADDR0[0] bit must match the I2C_CNFG[7] bit. For 10-bit mode, the address is written in I2C_CMD_ADDR0[9:0], and I2C_CNFG[7] indicates the read/write transaction.

33.5.4 I2C_I2C_CMD_DATA1_0

I2C Controller Data 1: Transmit/Receive

The four least significant bytes of data to be transmitted are loaded into this register when the I2C Master is in Write mode.

The four least significant bytes of data are read through this register when the I2C Master is in Read mode.

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

Bit	Reset	Description
31:24	0x0	DATA4: Fourth data byte to be sent/received
23:16	0x0	DATA3: Third data byte to be sent/received
15:8	0x0	DATA2: Second data byte to be sent/received
7:0	0x0	DATA1: First data byte to be sent/received

33.5.5 I2C_I2C_CMD_DATA2_0

I2C Controller Data 2: Transmit/Receive

The four most significant bytes of data to be transmitted are loaded into the register when the I2C Master is in Write mode.

The four most significant bytes of data are read through this register when the I2C Master is in Read mode.

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DATA8: Eighth data byte to be sent/received
23:16	0x0	DATA7: Seventh data byte to be sent/received
15:8	0x0	DATA6: Sixth data byte to be sent/received
7:0	0x0	DATA5: Fifth data byte to be sent/received

33.5.6 I2C_I2C_STATUS_0

I2C Controller Status (Master)

I2C_STATUS gives the status of the I2C Master operation.

Offset: 0x1c | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8	X	BUSY: 0 = NOT_BUSY 1 = BUSY
7:4	X	CMD2_STAT: Slave 2 status: Transaction for Slave 2 for byte x failed. x is 'h0 to 'ha. All other combinations are invalid. 0 = SL2_XFER_SUCCESSFUL 1 = SL2_NOACK_FOR_BYTE1 2 = SL2_NOACK_FOR_BYTE2 3 = SL2_NOACK_FOR_BYTE3 4 = SL2_NOACK_FOR_BYTE4 5 = SL2_NOACK_FOR_BYTE5 6 = SL2_NOACK_FOR_BYTE6 7 = SL2_NOACK_FOR_BYTE7 8 = SL2_NOACK_FOR_BYTE8 9 = SL2_NOACK_FOR_BYTE9 10 = SL2_NOACK_FOR_BYTE10
3:0	X	CMD1_STAT: Slave 1 status: Transaction for Slave 1 for byte x failed. x is 'h0 to 'ha. All other combinations are invalid. 0 = SL1_XFER_SUCCESSFUL 1 = SL1_NOACK_FOR_BYTE1 2 = SL1_NOACK_FOR_BYTE2 3 = SL1_NOACK_FOR_BYTE3 4 = SL1_NOACK_FOR_BYTE4 5 = SL1_NOACK_FOR_BYTE5 6 = SL1_NOACK_FOR_BYTE6 7 = SL1_NOACK_FOR_BYTE7 8 = SL1_NOACK_FOR_BYTE8 9 = SL1_NOACK_FOR_BYTE9 10 = SL1_NOACK_FOR_BYTE10

33.5.7 I2C_I2C_SL_CNFG_0

I2C Controller Configuration (Slave)

I2C_SL_CNFG register is used to configure, enable mode of slave ACK.

Enable mode of the slave response to the general call address.

The register should be programmed when the I^2C controller is configured as a slave.

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxx00000000000000000100)

Bit	Reset	Description
20	0x0	FIFO_XFER_EN: If this bit is disabled, data is always communicated via the I2C_SL_RCVD register. If enabled, data is communicated through FIFOs 0 = DISABLE 1 = ENABLE
19:8	0x0	BUFFER_SIZE: Payload size in bytes
7	0x0	ACK_LAST_BYTE_VALID: Acknowledge the last byte valid (write-only). This bit qualifies the ACK_LAST_BYTE field. 0 = DISABLE 1 = ENABLE
6	0x0	ACK_LAST_BYTE: Acknowledge the last byte. 0 = DISABLE 1 = ENABLE
5	0x0	ACK_WITHHOLD_EN: ACK Withhold Feature Enable 0 = DISABLE 1 = ENABLE
4	0x0	PKT_MODE_EN: Packet Mode Enable 0 = DISABLE 1 = ENABLE
3	0x0	ENABLE_SL: By writing zero to this field, the slave can be turned off 0 = DISABLE 1 = ENABLE
2	0x1	NEWSL: New Slave. 1 - Use new slave. 0 = DISABLE 1 = ENABLE
1	0x0	NACK: Disable Slave ACK. 1 – The slave will not acknowledge reception of address or data byte. 0 = DISABLE 1 = ENABLE
0	0x0	RESP: Slave response to the general call address (zero address). 0 = DISABLE 1 = ENABLE

33.5.8 I2C_I2C_SL_RCVD_0

I2C Controller Slave Receive/Transmit Data (Slave)

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	SL_DATA: Slave Received data

33.5.9 I2C_I2C_SL_STATUS_0

I2C Controller Slave Status (Slave)

Offset: 0x28 | Read/Write: R/W | Reset: 0x0000XX0X (0bxxxxxxxxxxxxxxxxxxxxxxxxx000000x0)

Bit	R/W	Reset	Description
14:8	RO	X	HW_MSTR_ADR: Hardware master address received via general call addressing. This field is meaningful only if HW_MSTR_INT is set.
7	RW	0x0	HW_MSTR_INT: 1 = Interrupt has been generated by slave. Hardware Master Address is received after General Call Address. Received Hardware Master Address. 0 = No event.
6	RW	0x0	REPROG_SL: 1 = Interrupt has been generated by the slave after the General Call Address is 0x04. Reprogram slave address. 0 = No action.
5	RW	0x0	RST_SL: 1 = Interrupt has been generated by slave after the General Call Address is 0x06. Reset and reprogram slave address. 0 = No action.
4	RW	0x0	END_TRANS: 1 = Interrupt has been generated by slave. Transaction completed as indicated by stop/repeat start condition. 0 = No transaction occurred or transaction in progress.
3	RW	0x0	SL_IRQ: 1 = Interrupt has been generated by slave. 0 = No interrupt generated. 0 = UNSET 1 = SET
2	RW	0x0	RCVD: New Transaction Received status. 1 = Transaction occurred. 0 = No transaction occurred. 0 = NO_TRANSACTION_OCCURED 1 = TRANSACTION_OCCURED
1	RO	X	RNW: Slave Transaction status. 0 = WRITE 1 = READ
0	RW	0x0	ZA: Zero Address Status 1 = Yes, slave responded. 0 = No, slave did not respond. 0 = NO_SLAVE_RESPONSE 1 = SLAVE_RESPONSE

33.5.10 I2C_I2C_SL_ADDR1_0

I2C Controller Slave Address 1 Register (Slave)

Offset: 0x2c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	SL_ADDR1: For a 10-bit slave address, this field is the least significant 8 bits.
7:0	0x0	SL_ADDR0: For a 10-bit slave address, this field is the least significant 8 bits.

33.5.11 I2C_I2C_SL_ADDR2_0

I2C Controller Slave Address 2 Register (Slave)

Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxxxx000xxxxx000)

Bit	Reset	Description
16	0x0	SELECT_SLAVE: 0 = Use slave addr0. 1 = Use slave addr1.
10:9	0x0	SL1_ADDR_HI: In 7-bit address mode, these bits are don't care. In 10-bit address mode, they represent the 2 MSBs of the address.
8	0x0	SL1_VLD: 0 = 7-bit addressing. 0 = SEVEN_BIT_ADDR_MODE 1 = TEN_BIT_ADDR_MODE
2:1	0x0	SL_ADDR_HI: In 7-bit address mode, these bits are don't care. In 10-bit address mode, they represent the 2 MSBs of the address.
0	0x0	VLD: 0 = 7-bit addressing. 1 - 10 bit addressing. 0 = SEVEN_BIT_ADDR_MODE 1 = TEN_BIT_ADDR_MODE

33.5.12 I2C_I2C_TLOW_SEXT_0

I2C Controller SMBUS Timeout Thresholds

Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx00000000000000000000000000)

Bit	Reset	Description
27	0x0	RST_SL_ON_TIMEOUT: Reset Slave state machine on time-out
26	0x0	TLOW_MEXT_EN: Enable TLOW_MEXT counter
25	0x0	TLOW_SEXT_EN: Enable TLOW_SEXT counter
24	0x0	TIMEOUT_EN: Enable TIMEOUT counter
23:16	0x0	TLOW_MEXT: Cumulative clock low extend time (master device) accumulated over a byte transfer period in milliseconds (START to ACK, ACK to ACK, or ACK to STOP).
15:8	0x0	TLOW_SEXT: Cumulative clock low extend time (slave device) accumulated over a complete transfer (START till STOP)
7:0	0x0	TIMEOUT: Clock low time-out period in milliseconds

33.5.13 I2C_I2C_SL_DELAY_COUNT_0

I2C Slave Controller Delay Count

Offset: 0x3c | Read/Write: R/W | Reset: 0x0000001e (0bxxxxxxxxxxxxxxxx0000000000011110)

Bit	Reset	Description
15:0	0x1e	SL_DELAY_COUNT: The value determines the timing between an address cycle and a subsequent data cycle or two consecutive data cycles on the bus. The I2C_SL_DELAY_COUNT is valid only when an internal slave is accessed. I2C_SL_DELAY_COUNT has to be programmed such that TIMING = T * DLY, where T is the period of the clock source selected for I2C, DLY is I2C_SL_DELAY_COUNT, and TIMING is the desired timing. A value of >= 1250 ns is recommended.

33.5.14 I2C_I2C_SL_INT_MASK_0

I2C Controller Slave Mask (Slave)

Offset: 0x40 | Read/Write: R/W | Reset: 0x000000fd (0bxxxxxxxxxxxxxxxxxxxxxxxxx111111x1)

Bit	Reset	Description
7	0x1	HW_MSTR_INT: 0 = DISABLE 1 = ENABLE
6	0x1	REPROG_SL: 0 = DISABLE 1 = ENABLE
5	0x1	RST_SL: 0 = DISABLE 1 = ENABLE
4	0x1	END_TRANS: 0 = DISABLE 1 = ENABLE
3	0x1	SL_IRQ: 0 = DISABLE 1 = ENABLE
2	0x1	RCVD: 0 = DISABLE 1 = ENABLE
0	0x1	ZA: 0 = DISABLE 1 = ENABLE

33.5.15 I2C_I2C_SL_INT_SOURCE_0

I2C Controller Slave Source (Slave)

Offset: 0x44 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7	X	HW_MSTR_INT: 0 = UNSET 1 = SET
6	X	REPROG_SL: 0 = UNSET 1 = SET
5	X	RST_SL: 0 = UNSET 1 = SET
4	X	END_TRANS: 0 = UNSET 1 = SET
3	X	SL_IRQ: 0 = UNSET 1 = SET
2	X	RCVD: 0 = UNSET 1 = SET

Bit	Reset	Description
0	X	ZA: 0 = UNSET 1 = SET

33.5.16 I2C_I2C_SL_INT_SET_0

I2C Controller Slave Source (Slave)

Offset: 0x48 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx000000x0)

Bit	Reset	Description
7	0x0	HW_MSTR_INT: 0 = UNSET 1 = SET
6	0x0	REPROG_SL: 0 = UNSET 1 = SET
5	0x0	RST_SL: 0 = UNSET 1 = SET
4	0x0	END_TRANS: 0 = UNSET 1 = SET
3	0x0	SL_IRQ: 0 = UNSET 1 = SET
2	0x0	RCVD: 0 = UNSET 1 = SET
0	0x0	ZA: 0 = UNSET 1 = SET

Note: Program the field PACKET_MODE_EN of I2C_CNFG register while working in packet mode.

The set of registers given below describe the interface for packet mode only. With packet mode interface changes, normal mode registers and the operation are not affected. The transition from normal mode to packet mode is suggested because packet mode allows:

- There is no restriction on the number of bytes before and the after the repeated start
- The number of bytes that can be transferred with a single command is not limited to 8.

Though a packet can contain 4 Kbytes, because any number of packets can be pushed into the FIFO, there is no limit on the number of bytes that can be transferred.

- The transactions to different slaves can be chained together with repeat start and there is no limit on the number of slaves it can address.

33.5.17 I2C_I2C_TX_PACKET_FIFO_0

A packet contains header and payload. The header size is variable and could vary from 2 to 5 words. For I2C, the header size is 3 words. The first two words of the header contain generic information. The third word contains I2C transaction-specific information. The payload contains actual data to be written to the slave. For read operations, the payload is nil. Thus, the packet contains a header only.

Offset: 0x50 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_PACKET: Software writes packets into this register. A packet may contain generic information

33.5.18 I2C_I2C_RX_FIFO_0

Header or I2C Specific Header or Data

Offset: 0x54 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RD_DATA: Software reads data from this register, causes a pop

33.5.19 I2C_PACKET_TRANSFER_STATUS_0

Offset: 0x58 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24	X	TRANSFER_COMPLETE: The packet transfer for which the last packet is set has been completed 0 = UNSET 1 = SET
23:16	X	TRANSFER_PKT_ID: The current packet ID for which the transaction is happening on the bus
15:4	X	TRANSFER_BYTENUM: The number of bytes transferred in the current packet
3	X	NOACK_FOR_ADDR: No ACK received for the address byte 0 = UNSET 1 = SET
2	X	NOACK_FOR_DATA: No ACK received for the data byte 0 = UNSET 1 = SET
1	X	ARB_LOST: Arbitration lost for the current byte 0 = UNSET 1 = SET
0	X	CONTROLLER_BUSY:1 = Controller is busy 0 = UNSET 1 = SET

33.5.20 I2C_FIFO_CONTROL_0

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:13	0x0	SLV_TX_FIFO_TRIG: Slave Transmit FIFO trigger level. 000 = 1 word DMA trigger is asserted when at least one word is empty in the FIFO 001 = 2 word DMA trigger is asserted when at least 2 words are empty in the FIFO

Bit	Reset	Description
12:10	0x0	SLV_RX_FIFO_TRIG: Slave Receive FIFO trigger level. 000 = 1 word DMA trigger is asserted when at least one word is full in the FIFO. 001 = 2 word DMA trigger is asserted when at least 2 words are full in the FIFO
9	0x0	SLV_TX_FIFO_FLUSH:1 = Flush the TX FIFO. Cleared after the FIFO is flushed 0 = UNSET 1 = SET
8	0x0	SLV_RX_FIFO_FLUSH:1 = Flush the RX FIFO. Cleared after the FIFO is flushed 0 = UNSET 1 = SET
7:5	0x0	TX_FIFO_TRIG: Transmit FIFO trigger level. 000 = 1 word DMA trigger is asserted when at least one word is empty in the FIFO. 001 = 2 word DMA trigger is asserted when at least 2 words are empty in the FIFO
4:2	0x0	RX_FIFO_TRIG: Receive FIFO trigger level. 000 = 1 word DMA trigger is asserted when at least one word is full in the FIFO. 001 = 2 word DMA trigger is asserted when at least 2 words are full in the FIFO
1	0x0	TX_FIFO_FLUSH:1 = Flush the TX FIFO. Cleared after the FIFO is flushed. 0 = UNSET 1 = SET
0	0x0	RX_FIFO_FLUSH:1 = Flush the RX FIFO. Cleared after the FIFO is flushed. 0 = UNSET 1 = SET

33.5.21 I2C_FIFO_STATUS_0

Offset: 0x60 | Read/Write: RO | Reset: 0x0XXX00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25	X	SLV_XFER_ERR_REASON: This bit describes the nature of the packet transfer error. It is meaningful only if PKT_XFER_ERR is set. 0 = The Master terminated the transaction before it was completed. 1 = The Master did not terminate the transaction when all bytes are transferred.
23:20	X	SLV_TX_FIFO_EMPTY_CNT: The number of slots that can be written to the slave TX FIFO. 0000 = tx_fifo full. 0001 = 1 slot empty. 0010 = 2 slots empty
19:16	X	SLV_RX_FIFO_FULL_CNT: The number of slots to be read from the Slave RX FIFO. 0000 = rx_fifo empty. 0001 = 1 slot full. 0010 = 2 slots full
7:4	X	TX_FIFO_EMPTY_CNT: The number of slots that can be written to the TX FIFO. 0000 = tx_fifo full. 0001 = 1 slot empty. 0010 = 2 slots empty
3:0	X	RX_FIFO_FULL_CNT: The number of slots to be read from the RX FIFO. 0000 = rx_fifo empty. 0001 = 1 slot full. 0010 = 2 slots full

33.5.22 I2C_INTERRUPT_MASK_REGISTER_0

Offset: 0x64 | Read/Write: R/W | Reset: 0x00000000 (0bxxx000000000xx00xxxx000000000000)

Bit	Reset	Description
28	0x0	SLV_ACK_WITHHELD_INT_EN: 0 = DISABLE 1 = ENABLE
27	0x0	SLV_RD2WR_INT_EN: 0 = DISABLE 1 = ENABLE
26	0x0	SLV_WR2RD_INT_EN: 0 = DISABLE 1 = ENABLE
25	0x0	SLV_PKT_XFER_ERR_INT_EN: 0 = DISABLE 1 = ENABLE
24	0x0	SLV_TX_BUFFER_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
23	0x0	SLV_RX_BUFFER_FILLED_INT_EN: 0 = DISABLE 1 = ENABLE
22	0x0	SLV_PACKET_XFER_COMPLETE_INT_EN: 0 = DISABLE 1 = ENABLE
21	0x0	SLV_TFIFO_OVF_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
20	0x0	SLV_RFIFO_UNF_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
17	0x0	SLV_TFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
16	0x0	SLV_RFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
11	0x0	BUS_CLEAR_DONE_INT_EN: 0 = DISABLE 1 = ENABLE
10	0x0	TLOW_MEXT_TIMEOUT_EN: 0 = DISABLE 1 = ENABLE
9	0x0	TLOW_SEXT_TIMEOUT_EN: 0 = DISABLE 1 = ENABLE
8	0x0	TIMEOUT_INT_EN: 0 = DISABLE 1 = ENABLE
7	0x0	PACKET_XFER_COMPLETE_INT_EN: 0 = DISABLE 1 = ENABLE
6	0x0	ALL_PACKETS_XFER_COMPLETE_INT_EN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	0x0	TFIFO_OVF_INT_EN: 0 = DISABLE 1 = ENABLE
4	0x0	RFIFO_UNF_INT_EN: 0 = DISABLE 1 = ENABLE
3	0x0	NOACK_INT_EN: 0 = DISABLE 1 = ENABLE
2	0x0	ARB_LOST_INT_EN: 0 = DISABLE 1 = ENABLE
1	0x0	TFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
0	0x0	RFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE

33.5.23 I2C_INTERRUPT_STATUS_REGISTER_0

This register indicates the status bit for which the interrupt is set. If the interrupt is set, write a 1 to clear it. However, the TFIFO_DATA_REQ and RFIFO_DATA_REQ fields depend on the FIFO trigger levels and cannot be cleared.

Slv_rd2wr indicates there is a switch from read to write by repeat start and the current read transaction needs to be closed and started with a write transaction. Similarly, slv_wr2rd indicates a switch from write to read.

Offset: 0x68 | Read/Write: R/W | Reset: 0x000X000X (0bxxx000000000xxxxxxxx0000000000xx)

Bit	R/W	Reset	Description
28	RW	0x0	SLV_ACK_WITHHELD: ACK is withheld, waiting for software explicit information about the ACK 0 = UNSET 1 = SET
27	RW	0x0	SLV_RD2WR:Transaction switching from read to write 0 = UNSET 1 = SET
26	RW	0x0	SLV_WR2RD:Transaction switching from write to read 0 = UNSET 1 = SET
25	RW	0x0	SLV_PKT_XFER_ERR: 0 = Request was successful. 1 = Error has occurred during packet transfer 0 = UNSET 1 = SET
24	RW	0x0	SLV_TX_BUFFER_REQ: Slave TX buffer is full 0 = UNSET 1 = SET
23	RW	0x0	SLV_RX_BUFFER_FILLED: Slave RX buffer is full 0 = UNSET 1 = SET
22	RW	0x0	SLV_PACKET_XFER_COMPLETE: Slave packet transfer complete 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
21	RW	0x0	SLV_TFIFO_OVF: Slave TX FIFO overflow 0 = UNSET 1 = SET
20	RW	0x0	SLV_RFIFO_UNF: Slave RX FIFO underflow 0 = UNSET 1 = SET
17	RO	X	SLV_TFIFO_DATA_REQ: Slave TX FIFO data request 0 = UNSET 1 = SET
16	RO	X	SLV_RFIFO_DATA_REQ: Slave RX FIFO data request 0 = UNSET 1 = SET
11	RW	0x0	BUS_CLEAR_DONE: Bus clear done status 0 = UNSET 1 = SET
10	RW	0x0	TLOW_MEXT_TIMEOUT: SMBUS mext time-out 0 = UNSET 1 = SET
9	RW	0x0	TLOW_SEXT_TIMEOUT:SMBUS sext time-out 0 = UNSET 1 = SET
8	RW	0x0	TIMEOUT:SMBUS time-out 0 = UNSET 1 = SET
7	RW	0x0	PACKET_XFER_COMPLETE: A packet has been transferred successfully. The TRANSFER_PKT_ID field can be used to know the current byte under transfer. This bit can be masked by the IE field in the I2C specific header. 0 = UNSET 1 = SET
6	RW	0x0	ALL_PACKETS_XFER_COMPLETE: All packets transferred successfully 0 = UNSET 1 = SET
5	RW	0x0	TFIFO_OVF: TX FIFO overflow 0 = UNSET 1 = SET
4	RW	0x0	RFIFO_UNF: RX FIFO underflow 0 = UNSET 1 = SET
3	RW	0x0	NOACK: No ACK from slave 0 = UNSET 1 = SET
2	RW	0x0	ARB_LOST: Arbitration lost 0 = UNSET 1 = SET
1	RO	X	TFIFO_DATA_REQ: TX FIFO data request 0 = UNSET 1 = SET
0	RO	X	RFIFO_DATA_REQ: RX FIFO data request 0 = UNSET 1 = SET

33.5.24 I2C_I2C_CLK_DIVISOR_REGISTER_0

The divisor values (N) must be programmed so that:

- SCL frequency (Standard/Fast/FM+ modes) = ClkSourceFreq/((tlow+thigh+3)*(N+1)) for lower values of N, up to 3
- SCL frequency (Standard/Fast/FM+ modes) = ClkSourceFreq/((tlow+thigh+2)*(N+1)) for higher values of N, above 3
- SCL frequency (High Speed mode) = ClkSourceFreq/((ths_low+ths_high+4)*(N+1)) for lower values of N, up to 4
- SCL frequency (High Speed mode) = ClkSourceFreq/((ths_low+ths_high+2)*(N+1)) for higher values of N, above 4

Offset: 0x6c | Read/Write: R/W | Reset: 0x00190001 (0b00000000000110010000000000000001)

Bit	Reset	Description
31:16	0x19	I2C_CLK_DIVISOR_STD_FAST_MODE: N = divide by n+1
15:0	0x1	I2C_CLK_DIVISOR_HSMODE: N = divide by n+1

33.5.25 I2C_I2C_INTERRUPT_SOURCE_REGISTER_0

This read-only register returns the AND of the Interrupt Source and Interrupt Mask registers.

Offset: 0x70 | Read/Write: RO | Reset: 0xXXXX0XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28	X	SLV_ACK_WITHHELD: 0 = UNSET 1 = SET
27	X	SLV_RD2WR: 0 = UNSET 1 = SET
26	X	SLV_WR2RD: 0 = UNSET 1 = SET
25	X	SLV_PKT_XFER_ERR: Error occurred during a slave transfer 0 = UNSET 1 = SET
24	X	SLV_TX_BUFFER_REQ: Slave TX buffer is full 0 = UNSET 1 = SET
23	X	SLV_RX_BUFFER_FILLED: Slave RX buffer is full 0 = UNSET 1 = SET
22	X	SLV_PACKET_XFER_COMPLETE: Slave packet transfer complete 0 = UNSET 1 = SET
21	X	SLV_TFIFO_OVF: Slave TX FIFO overflow 0 = UNSET 1 = SET
20	X	SLV_RFIFO_UNF: Slave RX FIFO underflow 0 = UNSET 1 = SET
17	X	SLV_TFIFO_DATA_REQ: Slave TX FIFO data request 0 = UNSET 1 = SET
16	X	SLV_RFIFO_DATA_REQ: Slave RX FIFO data request 0 = UNSET 1 = SET

Bit	Reset	Description
11	X	BUS_CLEAR_DONE: Bus clear done 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
8	X	TIMEOUT: SMBUS time-out 0 = UNSET 1 = SET
7	X	PACKET_XFER_COMPLETE: Packet transferred successfully 0 = UNSET 1 = SET
6	X	ALL_PACKETS_XFER_COMPLETE: All the packets transferred successfully 0 = UNSET 1 = SET
5	X	TFIFO_OVF: TX FIFO overflow 0 = UNSET 1 = SET
4	X	RFIFO_UNF: RX FIFO underflow 0 = UNSET 1 = SET
3	X	NOACK: No ACK from slave 0 = UNSET 1 = SET
2	X	ARB_LOST: Arbitration lost 0 = UNSET 1 = SET
1	X	TFIFO_DATA_REQ: TX FIFO data request 0 = UNSET 1 = SET
0	X	RFIFO_DATA_REQ: RX FIFO data request 0 = UNSET 1 = SET

33.5.26 I2C_I2C_INTERRUPT_SET_REGISTER_0

This write-only register can be used to set the interrupt status bit. A write to this register causes the bits in the status register to be set if the corresponding bit in the write data is 1'b1. A read always returns 'h0.

Offset: 0x74 | Read/Write: R/W | Reset: 0x00000000 (0bxxx000000000xxxxxxxx0000000000xx)

Bit	Reset	Description
28	0x0	SLV_ACK_WITHHELD: 0 = UNSET 1 = SET
27	0x0	SLV_RD2WR: 0 = UNSET 1 = SET
26	0x0	SLV_WR2RD: 0 = UNSET 1 = SET

Bit	Reset	Description
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
20	0x0	SLV_RFIFO_UNF: Slave RX FIFO underflow 0 = UNSET 1 = SET
11	0x0	BUS_CLEAR_DONE: Bus clear done 0 = UNSET 1 = SET
10	0x0	TLOW_MEXT_TIMEOUT:SMBUS mext time-out 0 = UNSET 1 = SET
9	0x0	TLOW_SEXT_TIMEOUT:SMBUS sext time-out 0 = UNSET 1 = SET
8	0x0	TIMEOUT:SMBUS time-out 0 = UNSET 1 = SET
7	0x0	PACKET_XFER_COMPLETE: Packet transferred successfully 0 = UNSET 1 = SET
6	0x0	ALL_PACKETS_XFER_COMPLETE: All the packets transferred successfully 0 = UNSET 1 = SET
5	0x0	TFIFO_OVF: TX FIFO overflow 0 = UNSET 1 = SET
4	0x0	RFIFO_UNF: RX FIFO underflow 0 = UNSET 1 = SET
3	0x0	NOACK: No ACK from slave 0 = UNSET 1 = SET
2	0x0	ARB_LOST: Arbitration lost 0 = UNSET 1 = SET

33.5.27 I2C_I2C_SLV_TX_PACKET_FIFO_0

Offset: 0x78 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_PACKET: Software writes packets into this register. A packet may contain generic information.

33.5.28 I2C_I2C_SLV_RX_FIFO_0

Header or I2C Specific Header or Data

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

Bit	Reset	Description
31:0	X	RD_DATA: Software reads data from this register, causes a pop.

33.5.29 I2C_I2C_SLV_PACKET_STATUS_0

Offset: 0x80 | Read/Write: RO | Reset: 0x0XXXXX0 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
25	X	ACK_WITHHELD: Indicates that an ACK is withheld for the last byte and the slave is waiting for the host to explicitly command the slave to acknowledge the last byte. 0 = Bus is released. 1 = ACK is withheld
24	X	TRANSFER_COMPLETE: All the packets have been transferred successfully
23:16	X	TRANSFER_PKT_ID: The current packet ID for which the transaction is happening on the bus
15:4	X	TRANSFER_BYTENUM: The number of bytes transferred in the current packet

33.5.30 I2C_I2C_BUS_CLEAR_CONFIG_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00090004 (0bxxxxxxxx00001001xxxxxxxxxxxxx100)

Bit	Reset	Description
23:16	0x9	BC_SCLK_THRESHOLD: send the clock pulses until this threshold is met
2	0x1	BC_STOP_COND: 0 = NO_STOP : Do not send a stop condition at the end of the bus clear operation 1 = STOP : Send a stop condition at the end of the bus clear operation
1	0x0	BC_TERMINATE: 0 = THRESHOLD: Irrespective of SDA release status during bus clear, terminate the bus clear only after the threshold is reached. 1 = IMMEDIATE: Terminate the bus clear operation immediately when SDA is released or threshold count is reached, whichever is earlier
0	0x0	BC_ENABLE: Starts bus clear operation. Hardware auto-clears this bit upon bus clear transaction completion

33.5.31 I2C_I2C_BUS_CLEAR_STATUS_0

Offset: 0x88 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	X	BC_STATUS: 0 = NOT_CLEARED: Indicates SDA is not released by the slave. Its status is still low.

Bit	Reset	Description
		1 = CLEARED: SDA is released

33.5.32 I2C_I2C_CONFIG_LOAD_0

Spare Register

Offset: 0x8c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	TIMEOUT_CONFIG_LOAD: This bit loads the timeout configuration from the pclk domain to the receive (i2c_slow_clk) domain. Software has to set this bit finally after doing the required registers configuration for the logic to take the updates. Once the internal update is done, hardware auto clears this bit. Since the hardware would be busy with internal update, software should not write again until this bit is cleared by hardware. 0 = DISABLE 1 = ENABLE
1	0x0	SLV_CONFIG_LOAD: This bit loads the slave configuration from the pclk domain to the receive (i2c_clk) domain. Software has to set this bit finally after doing the required registers configuration for the slave controller to take updates. Once the internal update is done, hardware auto clears this bit. Since the hardware would be busy with internal update, software should not write again until this bit is cleared by hardware. 0 = DISABLE 1 = ENABLE
0	0x0	MSTR_CONFIG_LOAD: This bit loads the master configuration from the pclk domain to the receive (i2c_clk) domain. Software has to set this bit finally after doing the required registers configuration like I2C_I2C_CNFG_0 bit fields etc. Once the internal update is done, hardware auto clears this bit. Since the hardware would be busy with internal update, software should not write again until this bit is cleared by hardware. 0 = DISABLE 1 = ENABLE

33.5.33 I2C_I2C_INTERFACE_TIMING_0_0

Register for Standard/Fast/FM+ mode timing

Offset: 0x94 | Read/Write: R/W | Reset: 0x00000204 (0bxxxxxxxxxxxxxxxxxxx000010xx000100)

Bit	Reset	Description
13:8	0x2	THIGH: High period of the SCL clock
5:0	0x4	TLOW: Low period of the SCL clock

33.5.34 I2C_I2C_INTERFACE_TIMING_1_0

Register for Standard/Fast/FM+ mode timing

Offset: 0x98 | Read/Write: R/W | Reset: 0x04070404 (0bxx000100xx000111xx000100xx000100)

Bit	Reset	Description
29:24	0x4	TBUF: Bus free time between STOP and START conditions
21:16	0x7	TSU_STO: Setup time for STOP condition
13:8	0x4	THD_STA: Hold time for a (repeated) START condition
5:0	0x4	TSU_STA: Setup time for a Repeated START condition

33.5.35 I2C_I2C_HS_INTERFACE_TIMING_0_0

I2C interface timing register for HS mode transfers. Because HS master code is sent in Std/FM/FM+ modes, Standard/Fast/FM+ timing registers need to be programmed along with HS timing registers.

Offset: 0x9c | Read/Write: R/W | Reset: 0x00000308 (0bxxxxxxxxxxxxxxxxxxx000011xx001000)

Bit	Reset	Description
13:8	0x3	HS_THIGH: High period of the SCL clock
5:0	0x8	HS_TLOW: Low period of the SCL clock

33.5.36 I2C_I2C_HS_INTERFACE_TIMING_1_0

I2C interface timing register for HS mode transfers. Because HS master code is sent in Std/FM/FM+ modes, Standard/Fast/FM+ timing registers need to be programmed along with HS timing registers.

Offset: 0xa0 | Read/Write: R/W | Reset: 0x000b0b0b (0bxxxxxxxxxx001011xx001011xx001011)

Bit	Reset	Description
21:16	0xb	HS_TSU_STO: Setup time for STOP condition
13:8	0xb	HS_THD_STA: Hold time for a (repeated) START condition
5:0	0xb	HS_TSU_STA: Setup time for a Repeated START condition

34.0 UART AND VFIR CONTROLLER

There are four Universal Asynchronous Receiver/Transmitters (UARTs) built into the Tegra® K1 devices. These UARTs support both 16450 and 16550 compatible modes. VFIR functionality is also supported in UART B.

34.1 Functional Description

34.1.1 UARTs A through D

All UARTs provide serial data synchronization and data conversion (parallel-to-serial and serial-to-parallel) for both receiver and transmitter sections. Synchronization for serial data stream is accomplished by adding start and stop bits to the transmit data to form a data character. Data integrity is accomplished by attaching a parity bit to the data character. The parity bit can be checked by the receiver for any transmission bit errors.

The COM-HOST interface is fully programmable through an 8-bit CPU interface. The registers are 32-bit word aligned. The interface supports word lengths from five to eight bits, an optional parity bit, and one or two stop bits. If enabled, parity can be odd, even, or forced to a defined state. Interrupts can be generated from any of 10 sources.

The UARTs support both 16450 and 16550 compatible modes. The default mode is 16450. This mode provides independent 16-byte FIFOs for transmit and receive operations and is selected by the FIFO control register. It also includes a 16-bit programmable baud rate generator and an 8-bit scratch register, 8 modem control lines, and 2 DMA handshake lines that are used to indicate when the FIFOs are ready to transfer data to the CPU.

The UARTs support a device clock of up to 72 MHz. The maximum baud rate is 16 clock cycles per symbol or 4.5 Megabits per second.

UART B is identical to UART A with the exception that only two signals are connected to its pins. UART A, C, and D are identical.

34.1.2 VFIR Controller

The VFIR controller in the Tegra K1 device implements the control and data sections of the IrDA Physical layer version 1.4, handling handshaking and basic networking between devices, and interfaces to an external Infrared transceiver.

This controller implements three distinct speed ranges and protocols: 2400 to 115,200 bits per second (Serial IR or SIR), 4 Megabits per second (called Fast Infrared or FIR), and 16 Megabits per second (called Very Fast Infrared or VFIR). Initial communication starts at 9600 bits per second, and negotiations proceed either faster or slower as the link strength allows. Each of the three speed ranges has different communication protocols and encoding schemes.

The following table lists the IrDA supported protocols.

Table 139: Protocols Supported by the IrDA

Signaling Rate (bps)	IrDA Name	Frame Wrapper and Encoding	IrDA Status	Tegra K1 Device Support?
9.6 K	SIR	Async RZI (3/16)	Mandatory	Yes
2400 - 115.2K	SIR	Async RZI (3/16)	Recommended	Yes
576.0 K	MIR	Sync HDLC RZI (4/16)	Optional	No
1.152 M	MIR	Sync HDLC RZI (4/16)	Optional	No
4.0 M	FIR	Sync 4PPM	Optional	Yes

Signaling Rate (bps)	IrDA Name	Frame Wrapper and Encoding	IrDA Status	Tegra K1 Device Support?
16.0 M	VFIR	Sync HHH(1,13)	Optional	Yes

The VFIR controller can operate in three different encodings; each is implemented separately. In SIR mode, a pulse encoder/decoder is inserted in the transmit/receive path of a standard UART. FIR and VFIR modes have their own control blocks. The outputs of the three blocks are multiplexed based on the current operating mode (SIR, FIR, or VFIR), before they leave the Tegra K1 device processor. The input signals are demultiplexed in a similar fashion.

Note: The operating mode is selected with the Mode [2:0] bits of the VFIR.CTL register

Figure 121: UART Block Diagram with SIR Decoder and Encoder

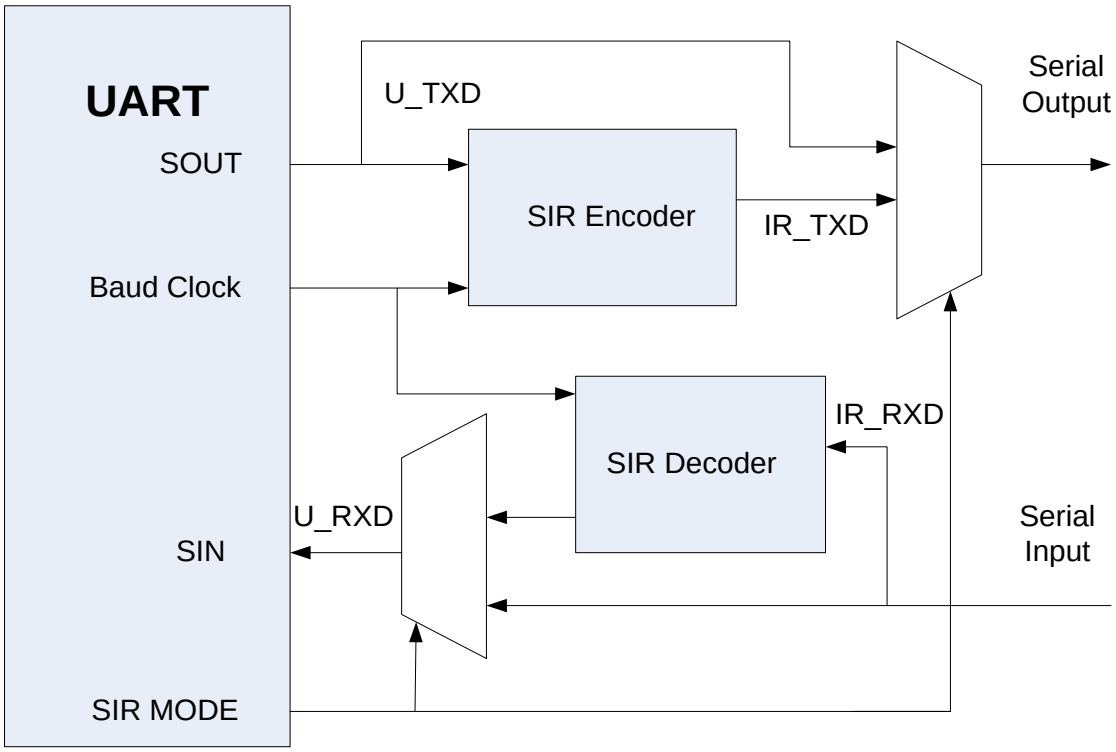
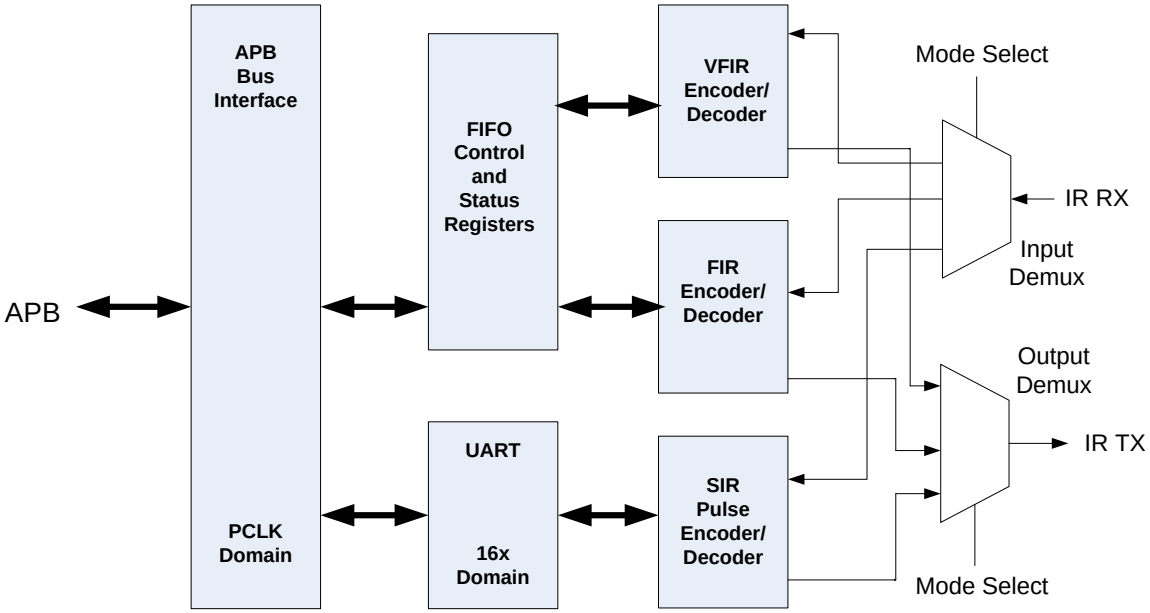


Figure 122: VFIR Controller Top-Level Block Diagram



34.1.3 Hardware Features

The features supported by UART A, UART B, UART C, and UART D are:

- Synchronization for the serial data stream with start and stop bits to transmit data to and from a data character
- Data integrity by attaching a parity bit to the data character
- The COM-HOST interface is fully programmable through an 8-bit CPU interface
- Support for word lengths from 5 to 8 bits, an optional parity bit, and 1 or 2 stop bits
- Support for both 16450- and 16550-compatible modes. The default mode is 16450.
- DMA capable for both TX and RX
- 8 bit x 36 deep TX FIFO
- 11 bit x 36 deep RX FIFO
- Auto sense baud detection
- Timeout interrupts to indicate if the incoming stream stopped
- Flow control support on RTS and CTS

The features supported by the VFIR controller are:

- Supports IrDA version 1.0 SIR protocol with maximum baud rate to 115.2 Kilobits per second
- Supports IrDA version 1.3 FIR protocol (4 Megabits per second)
- Supports IrDA version 1.4 VFIR protocol (16 Megabits per second)
- Programmable polarities on all the interface pins
- DMA capable for both TX and RX
- 32 bit x 16 deep FIFO
- Diagnostics for loopback and error insertion
- Interface control, such as transceiver signal strength, must be performed through GPIOs
- The maximum frequency of the device clock is 72 MHz

34.1.4 Hardware Signaling

All UARTs in the Tegra K1 device are implemented by a hardware block that supports modem control signals such as DTR, DSR, and DCD. UART A is, however, the only UART that allows those signals to be used externally.

For UART A, when the pin-mux is configured for 4- or 2-pin UART operation, the DSR and DCD input signals are tied low (active).

For UART B, the DTR output is routed back into the DSR and DCD inputs in hardware. This means that if software changes DTR state, the UART may raise an interrupt due to DSR/DCD changing state.

For UARTs C and D, the DSR and DCD inputs are tied to an inactive value.

Table 140: 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.

34.2 UART Programming Guidelines

34.2.1 16450 Mode Programming

1. Program pin mux settings to select a UART
2. Enable the UART clocks
3. For enabling internal loopback, program MCR[4] to 1
4. Program FCR[0] to 0
5. Enable interrupts in the IER register as needed
6. Write data into the THR register
7. Wait for a THR interrupt, if enabled, or poll for LSR[5]
8. During a receive, wait for an RBR interrupt or poll for LSR[0]
9. Read the UART.LSR register to clear interrupts

34.2.2 16550 Mode Programming

1. Program pin mux settings to select a UART
2. Enable the UART clocks
3. For enabling internal loopback, program MCR[4] to 1
4. Program FCR[0] to 1
5. Program trigger levels as required
6. Enable interrupts in the IER register as needed
7. Write data into the THR register
8. Wait for a THR interrupt, if enabled, or poll for LSR[5]
9. During a receive, wait for an RBR interrupt or poll for LSR[0]
10. Read the UART.LSR register to clear interrupts
11. The APB-DMA requestor numbers for UARTs A, B, and C are 8, 9, and 10, respectively

34.2.3 Transmitter and Receiver Holding Registers

The serial transmitter section consists of a Transmit Hold Register (THR) and Transmit Shift Register (TSR). The status of transmit hold register is provided in the Line Status Register (LSR). Writing to this register (THR) transfers the contents of data bus (D [7:0]) to the Transmit Holding Register whenever the Transmitter Holding Register or Transmitter Shift Register is empty. The Transmit Holding Register empty flag is set to 1 when the transmitter is empty or data is transferred to the transmit shift register. Note that a write operation is performed when the Transmit Holding Register empty flag is set.

The serial receiver section also contains an 8-bit Receiver Holding/Buffer Register (RBR). Receive data is removed from the UART and received by the processor by reading the RBR. The receiver contains a mechanism for preventing false starts as follows: On the falling edge of the start bit, the receiver internal counter starts to count 7.5 clocks (16x clock), which is the center of the start bit. The start bit is valid if the RX is still low at the mid-bit sample of the start bit. Verifying the start bit prevents the receiver from assembling a false data character due to a low going noise spike on the RX input. Receiver status codes are posted in the Line Status Register.

34.2.4 FIFO Interrupt Mode Operation

When the receive FIFO (FCR bit 0 = 1) and receive interrupts (IER bit 0 = 1) are enabled, a receiver interrupt occurs as follows:

- The receive data available interrupts are issued to the CPU when the FIFO has reached its programmed trigger level; it is cleared as soon as the FIFO drops below its programmed trigger level.
- The ISR receive data available indication also occurs when the FIFO trigger level is reached, and like the interrupt, it is cleared when the FIFO drops below the trigger level.
- The data ready bit (LSR bit 0) is set as soon as a character is transferred from the shift register to the receiver FIFO. It is reset when the FIFO is empty.

34.2.5 FIFO Polled Mode Operation

When FCR bit 0 = 1; clearing IER bits [3:0] to zero puts the UART in the FIFO polled mode of operation. Since the receiver and transmitter are controlled separately, either one or both can be in the polled mode operation by using the line status register as follows:

- LSR bit 0 is set as long as there is one byte in the receive FIFO
- LST bits [4:1] specifies which error(s) have occurred
- LSR bit 5 indicates when the transmit FIFO is empty
- LSR bit 6 indicates when both transmit FIFO and transmit shift registers are empty
- LSR bit 7 indicates when there are any errors in the receive FIFO
- LSR bit 8 indicates when the transmit FIFO is full

The UART requires a two-step FIFO enable operation in order to enable receive trigger levels.

34.2.6 Programmable Baud Rate Generator

The UART contains a programmable baud rate generator that can take the UART clock input and divide it by any divisor from 1 to 2^{16}-1. Refer to the Clocking and Reset Controller section for definitions of the UART clocks. The output frequency of baudout is equal to 16X the transmission baud rate (Baudout=16 X baud rate). Customized baud rates are achieved by selecting proper divisor values for the MSB and LSB bits of the baud rate generator.

34.2.7 Enable Register (IER)

There is an Interrupt Enable Register for each UART (UART A Interrupt Enable and UART B Interrupt Enable). The Interrupt Enable Register(s) masks the incoming interrupts from the receiver ready, transmitter empty, line status, and modem status registers to the INT output pin.

34.2.8 Interrupt Identification Register (IIR)

The UART provides four level prioritized interrupt conditions to minimize software overhead during data character transfers. The Interrupt Identification Register (IIR) provides the source of the interrupt in a prioritized manner.

During the read cycle, the 16550 provides the highest interrupt level to be serviced by the CPU. No other interrupts are acknowledged until the particular interrupt is serviced. The prioritized interrupt levels are shown in the following table. The Receive Data Time-out mode is enabled when the UART is operating in the FIFO mode.

Receive time-out does not occur if receive FIFO is empty. The time-out counter is reset at the center of each stop bit received or each time the Receive Holding Register is read. The actual time out value is:

- T (Time out length in bits) = 4 X P (Programmed work length) + 12

To convert the time-out value to a character value, divide this number to its complete word length + parity (if used) + number of stop bits and start bit.

Example

If you program the word length = 7 with no parity and one stop bit, the time-out is:

- T = 4X7 (programmed word length) + 12 = 40 bits.
- Character time = 40/9
- [(Programmed word length = 7) + (stop bit = 1) + (start bit = 1)] =4.4 characters

Table 141 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)

34.2.9 FIFO Control Register (FCR) Modes

The FIFO control register is used to enable the FIFOs, clear the FIFOs, set the receiver FIFO trigger level, and select the type of DMA signaling. The operation of the FCR in the four DMA modes is given below.

34.2.9.1 Transmit Operation in DMA Mode 0 or Mode 1

When the UART is in the 16450 mode (FCR bit 0 = 0) or in the FIFO mode (FCR bit 0 = 1, FCR bit 3 = 0) and when there are no characters in the transmit FIFO or transmit holding register, the TXRDY* pin goes low. Once active, the TXRDY* pin goes high (inactive) after the first character is loaded into the Transmit Holding Register.

34.2.9.2 Receive Operation in DMA Mode 0

When the UART is in 16450 mode (FCR bit 0 = 0) or in the FIFO mode (FCR bit 0 = 1,FCR bit 3 = 0) and there is at least 1 character in the receive FIFO, the RXRDY* pin goes low. Once active, the RXRDY* pin goes high (inactive) when there are no more characters in the receiver.

34.2.9.3 Receive Operation in DMA Mode 1

When the UART is in the FIFO mode (FCR bit 0 = 1, FCR bit 3 = 1) and the trigger level or the timeout has been reached, the RXRDY* pin goes low. Once it is activated, it goes high (inactive) when there are no more characters in the FIFO.

34.2.10 Line Control Register (LCR)

The Line Control Register is used to specify the asynchronous data communication format. The word length, stop bits, and parity can be selected by writing appropriate bits in this register.

34.2.11 Modem Control Register (MCR)

This register controls the interface with the modem or a peripheral device (RS232).

Loopback Mode

If MCR[4] = 1, the loopback mode is enabled, and the following occurs:

The transmitter output (TX) is set high (Mark condition), and the receiver input (RX), CTS, DSR, CD, and RI are disabled. Internally, the transmitter output is connected to the receiver input and DTR, RTS, OP1, and OP2 are connected to modem control inputs.

In this mode, the receiver and transmitter interrupts are fully operational, but the interrupt sources are now the lower four bits of the modem control register instead of the four modem control inputs. The interrupts are controlled by the IER register.

Table 142: Modem Control Register (MCR)

Bit	Name	Description
7	N/A	Not used. Set to 0 internally.
6	RTS_EN	1 = Enable Hardware Flow Control using RTS 0 = Disable Hardware Flow Control
5	CTS_EN	1 = Enable Hardware Flow Control using CTS 0 = Disable Hardware Flow Control
4	LOOP	0 = Normal operating mode. 1 = Enable local loop-back mode (diagnostic).
3	OUT2	nOUT2 polarity
2	OUT1	nOUT1 polarity
1	RTS	1 = Force RTS (Request to Send) low 0 = Force RTS to high
0	DTR	1 = Force DTR (Data Terminal Ready) to low 1 = Force to high

34.2.12 Line Status Register (LSR)

The Line Status Register provides the status of data transfer to the CPU.

34.2.13 Modem Status Register (MSR)

The modem status register provides the current state of the control lines from the modem or peripheral to the CPU. Four bits of this register are used to indicate the changed information. These bits are set to 1 whenever a control input from the modem changes state. They are set to 0 whenever the CPU reads the register.

34.2.14 Scratchpad Register (SR)

Eight bits of information can be stored in this register. Information in this register does not affect the operation of the device in any way.

34.2.15 Autobaud Sense Register (ASR)

The UART can automatically determine the correct baud divisor by using the Autobaud Sense Register (UART offsets 0x3C). The most significant bit of this register is the valid flag. A write to this register will clear the valid flag and enable the autobaud process. When the first RX edge occurs, a counter running at 24 MHz starts counting. When another rx_edge occurs, the "complete" flag is set, the value is frozen, and the Autobaud Sense Value register is updated with the count value. The low 20 bits of the ASR give the number of clocks within a single bit. Because the UART uses 16x oversampling, the resulting value needs to be adjusted by shifting right 4 bits, then loading the resulting count in the divisor latch of the UART. (In the code snippet below, the lower 4 bits are rounded to give slightly greater accuracy.)

Because the speed determination is made by measuring the start bit, special characters must be sent by the transmitting UART to guarantee that the next character after the start bit is a 1. Since bit 0 (rightmost bit) is sent first, the ASCII Carriage Return character (CR) is sufficient to enable proper speed sense.

The following code snippet returns the values for DLH and DLL in r9, r8:

```
       MOV     r0, #1  ; dummy write data
       STRB    r0, [r3, #U_ASR]; Start autobaud sense (r3 as uart_base)
; now poll the autobaud sense register MSB until Valid is true
wait4valid
       LDR     r2, [r3, #U_ASR]; Read ASR
       TST     r2, #0x80000000 ; the Valid bit (active high)
       BEQ     wait4valid
; autobaud sense check complete...
; r2 as number of 1x clocks in one bit time
; representing the number of 24MHz clocks in the start bit
; Since this represents 16 of the baud (16x) clocks, we
; will be dividing by 16 to get baud divisor, but first
; round to nearest by adding 8 before the divide:
       ADD     r4, r2, #8 ; add 1/2 resolution
       MOV     r6, r4, LSR #4 ; divide by 16... R6 will have total divisor
       AND     r8, r6, #0xFF  ; copy DLL to r8
       MOV     r9, r6, LSR #8 ; copy DLM to r9
       AND     r9, r9, #0xff ; mask upper bytes
```

34.2.16 Baud Rate Generator

The following table given below is a divisor table for the baud rate, assuming the oscillator is 24.000 MHz.

Table 143: 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

34.2.17 Power-up Defaults

The Power-Up defaults are defined below:

- 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

34.3 VFIR Programming Guidelines

34.3.1 IRDA (FIR/VFIR)

The IRDA module includes a fully functional 16550 compatible UART for implementing the SIR protocol. The UART pins will be selected automatically when the VFIR.CTL Mode bits are set to UART or SIR mode. Set the UART IRDA.CSR register to SIR mode to convert the UART signals to SIR signals.

If the IRDA.CTL Mode bits are set the FIR or VFIR modes, the UART pins will be disabled, and the FIR/VFIR pins will be used.

34.3.2 APB Bus Interface to Control/Status/FIFO

The Control and Status registers, and the FIFO reside on the APB bus. The processor programs the Control registers to begin transactions, and can read progress and error status from the Status Register. FIFO access is normally performed via the APB_DMA module.

34.3.3 FIR Mode Programming

1. Set the CLK_SOURCE_VFIR for 48 MHz and enable the UART_B Clock
2. Program IR.TX and IR.RX pin-mux settings to select IR (UART B)
3. Set the VFIR.CTL Mode to binary 100 (FIR)
4. FIR mode is half-duplex. Set the VFIR.CTL Transmit bit to '1' for transmits, '0' for receives
5. Read the VFIR.STS register to clear inputs
6. Program the APB_DMA to transfer data to/from the VFIR module
7. Enable the DMA bit in the VFIR.CTL register
8. Enable interrupts as needed
9. Set the GO bit in the VFIR.CTL register to start the action
10. Wait for DONE status or an interrupt which indicates that the transaction has completed
11. Check for errors when reading the VFIR Status and Interrupt Identification register
12. Clear the status bits by writing a '1' to them before starting the next packet.
13. For receive operations, the Incoming Frame Data Length (IFDL) register will contain the number of bytes received. DMA should have read this many bytes from the FIFO.

34.3.4 VFIR Mode Programming

1. Set the CLK_SOURCE_VFIR for 48MHz and enable the UART_B Clock
2. Program IR.TX and IR.RX pin mux settings to select IR (UART B)
3. Set the VFIR.CTL Mode to binary 101 (VFIR)
4. VFIR mode is half-duplex. Set the VFIR Control register Transmit bit to '1' for transmits, and '0' for receives

5. Read the VFIR.STS register to clear inputs
6. Program the APB_DMA to transfer data to/from the VFIR module
7. Enable the DMA bit in the VFIR.CTL register
8. Enable interrupts as needed
9. Set the GO bit in the VFIR.CTL register to start the action
10. Wait for DONE status or an interrupt which indicates that the transaction has completed
11. Check for errors when reading the VFIR Status and Interrupt Identification register
12. Clear the status bits by writing a '1' to them before starting the next packet.
13. For receive operations, the Incoming Frame Data Length (IFDL) register will contain the number of bytes received. DMA should have read this many bytes from the FIFO.

34.3.5 SIR Interface

The SIR (COM-SLAVE) interface consists of a UART, which provides serial data synchronization and parallel-to-serial and serial-to-parallel data conversion for both receiver and transmitter sections. Synchronization for the serial data stream is accomplished by adding start and stop bits to the transmit data to form a data character. Data integrity is accomplished by attaching a parity bit to the data character. The parity bit can be checked by the receiver for any transmission bit errors.

The SIR interface is fully programmable by an 8-bit CPU interface. The registers are 32-bit Word aligned. The interface supports word lengths from 5 to 8 bits, an optional parity bit, and 1 or 2 stop bits. If enabled, the parity can be odd, even, or forced to a defined state. Interrupts can be generated from any of 10 sources.

The UART supports both 16450 and 16550 compatible modes. The default mode is the 16450. The 16550 mode, which provides independent 16-byte FIFOs for transmits and receives, is selected via the FIFO control register. It also includes a 16-bit programmable baud rate generator and an 8-bit Scratch register, 8 modem control lines, and two DMA handshake lines that are used to indicate when the FIFOs are ready to transfer data to the CPU.

Table 144: Serial Bus Interface Signals

Signal Name	Direction	Description
UB3_RXD	I	Receiver Data Port
UB3_TXD	O	Transmit Data Port

34.3.6 SIR Pulse Encoder/Decoder

The UART transmit (TX) data is passed through this module prior to being muxed out to the IR transmitter. This module converts transmitted zeros into a 3/16 Return-To-Zero (RZ) pulse.

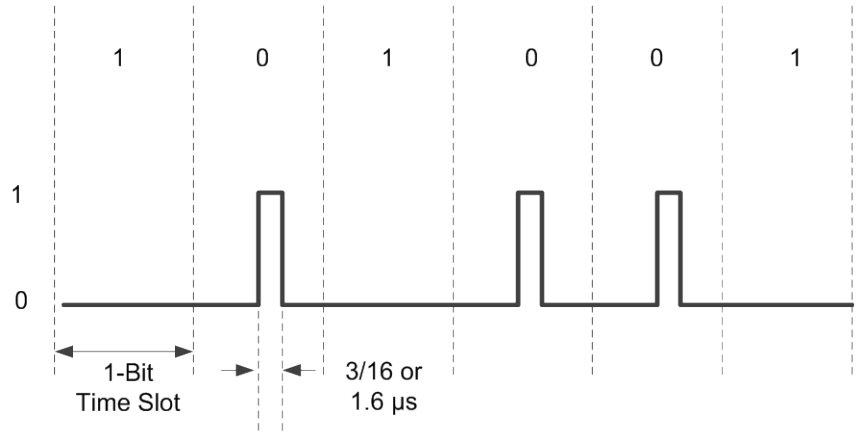
Similarly, received (RX) data is bit-synchronized using the 16X baud clock and the original serial data recovered from the IR bit stream. This data is then sent to the UART for reception.

The signal generated in SIR is as follows:

- On logic '1', the LED is off.
- On logic '0', a pulse is created starting the center of the bit time and lasting 3/16 of a bit time period or 1.6µs (3/16 bit times at 115.2 kbps) depending on the current settings.

The figure below displays the output for the input 101001 (in sending order) in SIR encoding.

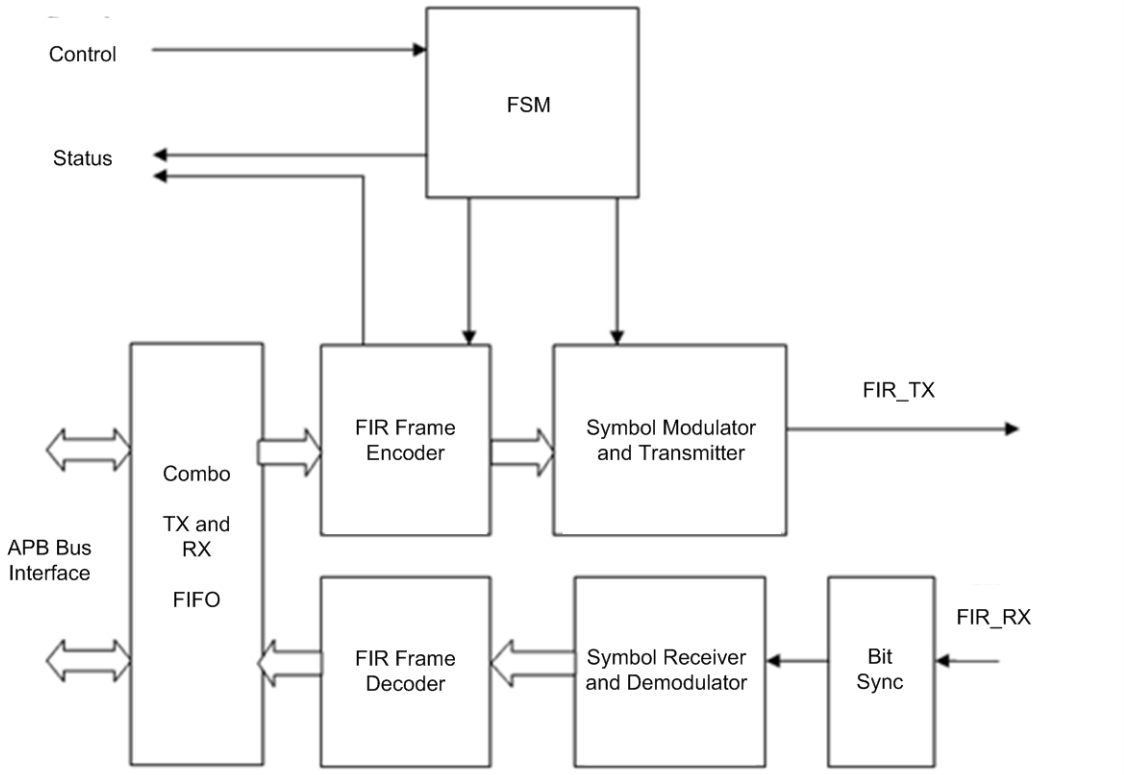
Figure 123: Output in Sending Order in SIR Encoding



34.3.7 FIR Encoder/Decoder

The figure below shows the micro-architecture for FIR and VFIR modes. Although the architecture is the same, the FIR and VFIR encoding schemes are very different.

Figure 124: Architecture for FIR and VFIR Modes



The Frame Level Encoder calculates CRC for the sent data, performs bit stuffing, creates start and stop sequences, and sends data down the line to the Data Shift Register. The Frame Level Decoder calculates the CRC on incoming data, performs bit de-stuffing, compares the calculated CRC to the received one, and reports any errors detected in the process.

The 1-symbol modulator creates the signal to be sent to the IR transceiver bit by bit as received from the UART in SIR mode, 2-bit signals in FIR mode, and 3-bit signals in VFIR mode.

FIR mode uses a Pulse Position Modulation code called 4PPM. The encoding sends one symbol every four time slices called "chips". Because there are four unique chip positions within each symbol in 4PPM, four independent symbols exist in which only one chip is logically a "one" while all other chips are logically a "zero." These four unique symbols are defined to be the only legal data symbols (DD) allowed in 4PPM. Each DD represents two bits of payload data, so that a byte of payload data can be represented by four DDs in sequence. The table below defines the chip pattern representation of the four unique DDs defined for 4PPM.

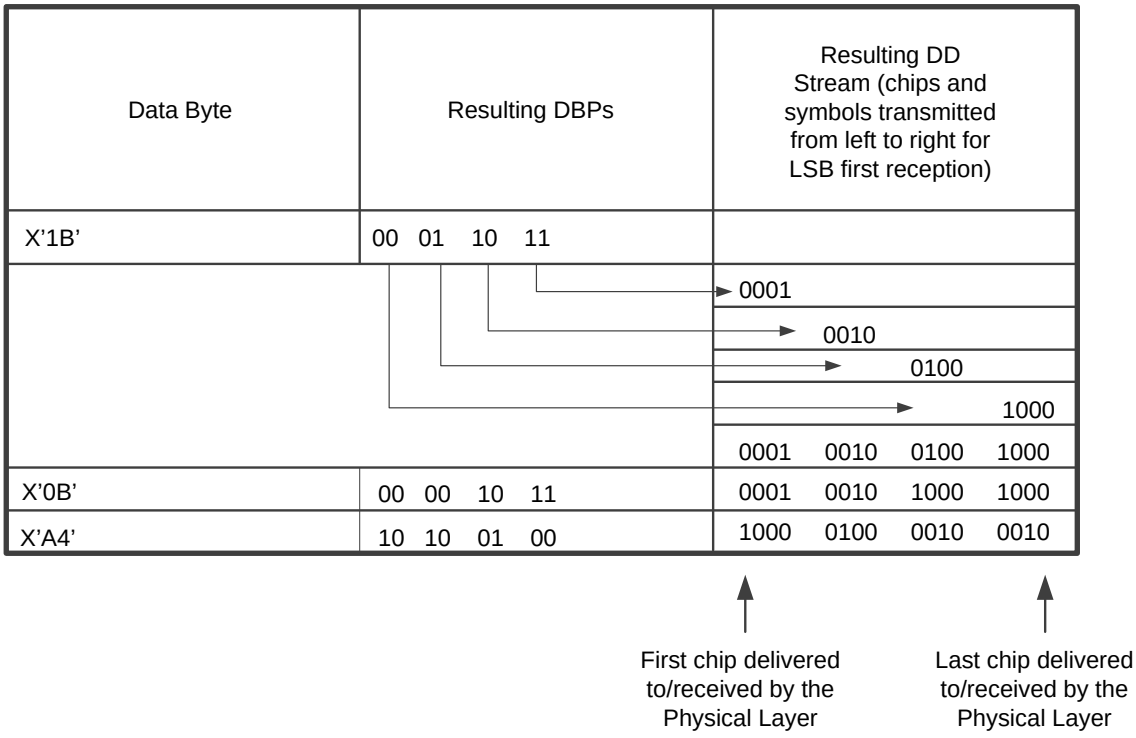
Table 145: Chip Pattern

Data Bit Pair	4PPM Data Symbol
00	1000
01	0100
10	0010
11	0001

A logical "1" represents a chip duration when the transmitting LED is emitting light, while a logical "0" represents a chip duration when the LED is off. Data encoding for transmission is done LSB first.

The figure below shows how various data bytes would be represented after encoding for transmission. In these examples transmission time increases from left to right so that chips and symbols farthest to the left are transmitted first.

Figure 125: Various Data Bytes After Encoding for Transmission



34.3.8 PPM Packet Format

For FIR 4PPM packets the data rate is 4 Megabits per second, and the signaling rate is 8.0 Mchips per second, where a chip is the smallest element of IrDA signaling. The packet format is defined below.

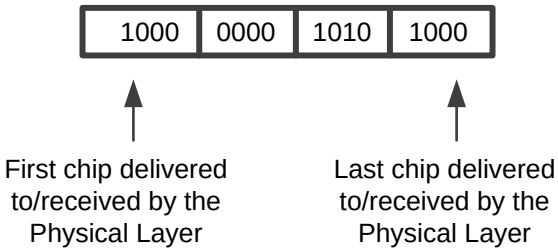
In this packet format, the payload data is encoded as described in the 4PPM encoding above, and the encoded symbols reside in the DD field. Maximum packet length is negotiated by the same mechanism as for the slower rates. The preamble field (PA) is used by the receiver to establish phase lock. During PA, the receiver begins to search for the start flag (STA) to establish symbol synchronization. If STA is received correctly, the receiver can begin to interpret the data symbols in the DD field. The receiver continues to receive and interpret data until the stop flag (STO) is recognized. STO indicates the end of a frame. The chip patterns and symbols for PA, STA, FCS field, and STO are defined below. Only complete packets that contain the entire format defined above are guaranteed to be decoded at the receiver (as for the lower rates, the information field, I, may be of zero length).

The 4PPM data encoding described above defines only the legal encoded payload data symbols. All other chip combinations are by definition illegal symbols for encoded payload data. Some of these illegal symbols are used in the definition of the preamble, start flag, and stop flag fields because they are unambiguously not data.

34.3.8.1 Preamble Field Definition

The preamble field (PA) consists of exactly 16 repeated transmissions of the following stream of symbols. In the PA field, transmission time increases from left to right so that chips and symbols on the left are transmitted first. See figure below.

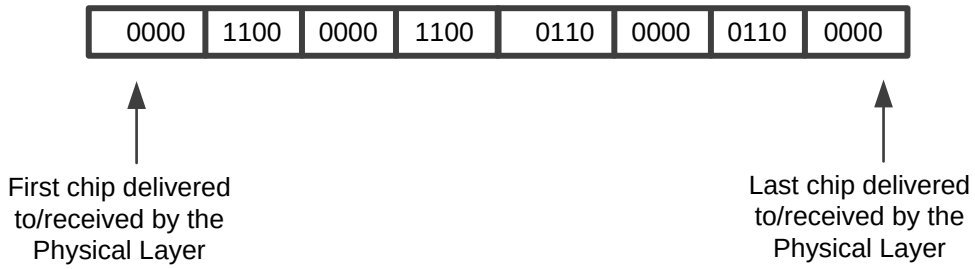
Figure 126: Preamble Field



34.3.8.2 Start Flag Definition

The start flag (STA) consists of exactly one transmission of the following stream of symbols. In the STA field, transmission time increases from left to right so that chips and symbols on the left are transmitted first. See figure below.

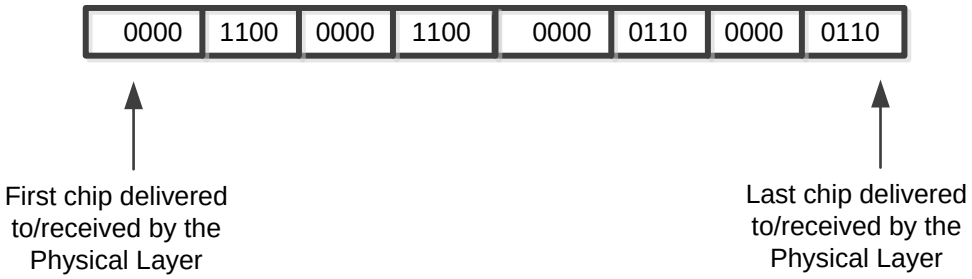
Figure 127: Start Flag



34.3.8.3 Stop Flag Definition

The stop flag (STO) consists of exactly one transmission of the following stream of symbols. In the STO field, transmission time increases from left to right so that chips and symbols on the left are transmitted first. See figure below.

Figure 128: Stop Flag



34.3.8.4 Frame Check Sequence Field Definition

The frame check sequence (FCS) field is a 32-bit field that contains a cyclic redundancy check (CRC) value. The CRC is a calculated, payload data dependent field, calculated before 4PPM encoding. It consists of the 4PPM encoded data resulting from the IEEE 802 CRC32 algorithm for cyclic redundancy check as applied to the payload data contained in the packet. The CRC32 polynomial is defined as follows:

$$CRC(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$$

The CRC32 calculated result for each packet is treated as four data bytes, and each byte is encoded in the same fashion as is payload data. Payload data bytes are input to this calculation in LSB first format. The 32-bit CRC register is preset to all "1's" prior to calculation of the CRC on the transmit data stream. When data has ended and the CRC is being shifted for transmission at the end of the packet, a "0" should be shifted in so that the CRC register becomes a virtual shift register.

Note: The inverse of the CRC register is what is shifted as defined in the polynomial.

34.3.8.5 Aborted Packets

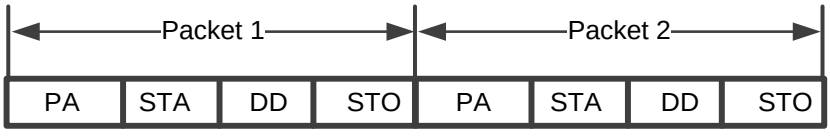
Receivers may only accept packets that have valid STA, DD, FCS, and STO fields as defined in the PPM Packet Format section. The PA field need not be valid in the received packet. All other packets are aborted and ignored.

Any packet may be aborted at any time after a valid STA but before transmission of a complete STO flag by two or more repeated transmissions of the illegal symbol "0000." Also, any packet may be aborted at any time after a valid STA by reception of any illegal symbol which is not part of a valid STO field.

34.3.8.6 Back-to-Back Packet Transmission

Back-to-back, or "brick-walled", packets are allowed, but each packet must be complete (i.e., containing PA, STA, DD, and STO fields). Brick-walled packets are illustrated in the figure below.

Figure 129: Back-to-Back (Brick-Walled) Packet Transmission



34.3.9 VFIR 16.0 Megabits per Second Rate

The 16.0 Megabits per second data transmission incorporates a new modulation code HHH (1,13) - low duty cycle, rate 2/3, (d,k) = (1, 13) run-length limited (RLL) code to achieve the specified data rate from a 24MHz reference clock. The HHH(1,13) code guarantees for at least one empty chip and at most 13 empty chips between chips containing pulses in the transmitted IR signal. The data transmission packet/frame is based on the 4.0 Megabits per second frame format with modifications introduced where necessary to accommodate the requirements that are specific to the new modulation code. The system

includes a simple scrambling/descrambling scheme to randomize the duty cycle statistics. The signaling rate of the 16.0 Megabits per second data rate is 24.0 Mchips per second, where a chip is the smallest element of IrDA signaling.

34.3.10 HHH (1, 13) Modulation Code

The HHH (1, 13) modulation code has the following salient features:

- Code Rate: 2/3 ,
- Maximal Duty Cycle: 1/3 (~33%) ,
- Average Duty Cycle: ~26% ,
- Minimal Duty Cycle: 1/12 (~8.3%) ,
- Run-Length Constraints: (d, k) = (1, 13) ,
- Longest Run of '10's: yyy'000'101'010'101'000'yyy ,
- Chip Rate @ Data Rate 16 Megabits per second: 24 Mchips per second ,
- System Clock @ Data Rate 16 Megabits per second: N X 12 MHz (where N ≥ 4)

The HHH(1,13) code is a Run Length Limited (RLL) code that provides both power efficiency and bandwidth efficiency at the high data rate. The signaling rate of the code is 24 Mchips per second allowing a rise and fall time of 19 ns. LED on time is further improved by having a 26% average duty cycle for random data. The lower duty cycle is achieved by scrambling the incoming data stream. The run length constraints (d, k) = (1, 13) ensure an inactive chip after each active chip, i.e. only single-chip-width pulses occur. This feature allows a source or a receiver to exhibit a long tail property. To take full advantage of the d = 1 feature of HHH(1, 13) in strong signal conditions, clock and data recovery circuitry ignores the level of the chip following an active chip and assumes these chips are inactive. The modulation code is enhanced with simple frame-synchronized scrambler/descrambler mechanisms as defined and described in later in this chapter.

34.3.10.1 HHH Data Encoding Definition

The encoding definition of the HHH (1, 13) code is provided by a state transition table described as follows:

Specific data pair D ≡ □D* = (d1, d2) arriving at the encoder input is first associated with a corresponding next state N □ N* This occurs as soon as the data D* bits have advanced into the positions of the internal data bits B1 = (b1, b2), i.e., when (b1, b2, b3, b4, b5, b6) is identical to (d1, d2, x, x, x, x). In a second step, during the next encoding cycle, the state S takes on the value of N*, i.e., S ≡ S* ← □N* so that S is now associated with (d1, d2). In the same cycle, the inner codeword C □C* 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)

Figure 130: Encoder Initialization

Present State: S = (s_1, s_2, s_3)	Next State/Internal Output: N = (n_1 n_2, n_3) / C = (c_1, c_2, c_3)							
	Internal Inputs: (b_1, b_2, b_3, b_4, b_5, b_6)							
	00xxxx	01xxxx	10xxxx	1100xx	1101xx	111011	1110(~~11~~)	1111xx
000	000/010	001/010	010/010	111/010	111/001	111/010	011/010	011/010
001	000/001	001/001	100/001	100/010	100/010	100/010	100/010	100/010
010	000/100	001/100	010/100	111/100	111/101	111/100	011/100	011/100
011	000/101	001/101	100/101	100/100	100/100	100/100	100/100	100/100
100	000/000	001/000	010/000	011/000	011/000	011/000	011/000	011/000
111	100/000	100/000	111/000	100/000	100/000	100/000	100/000	100/000

Note: Refer to the IrDA specification 1.4 for further details.

34.3.10.2 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 (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.

34.3.10.3 Preamble Field Definition

The transmitted PREAMBLE (PA) is constructed by concatenating ten times (10X) the 24-chip (1 µs) PREAMBLE PERIOD (PP), where

PP = '100'010'010'001'001'001'000'100'

to form the complete 240-chip (10 µs) preamble

PA = 'PP'PP'PP'PP'PP'PP'PP'PP'PP'PP'

The left-most/right-most chip of PP and PA, respectively, is transmitted first/last and a '1' in PP means an active chip (pulse) and a '0' means an empty chip (no pulse).

34.3.10.4 Start (STA) Flag Definition

The transmitted START (STA) delimiter is the 48-chip (2 µs) chip sequence

STA = '100'101'010'100'100'010'000'001'001'010'101'001'000'001'010'000'

The left-most/right-most chip of STA is transmitted first/last and a '1' in STA means an active chip (pulse) and a '0' means an empty chip (no pulse). The Start Flag Delimiter allows for packet synchronization. A delimiter detection circuit should declare a flag as having been found when there is a perfect match between the receiver chip stream and a particular delimiter. The Start and Stop delimiters contain a subsequence '1001010101001' that violates the HHH (1,13) code. This subsequence occurs twice in the Start Flag delimiter and never occurs within the main HHH code.

34.3.10.5 IrLAP Frame

The structure remains unchanged from that defined in the IrLAP Specification, Version 1.1. The content of the IrLAP frame is first scrambled and then encoded with HHH(1,13). Note that the 32 CRC bits for the IrLAP frame are calculated before the IrLAP frame is scrambled. For reference, the IrLAP frame as the following structure:

| Address (8 bits) | Control (8 bits) | Information (M times 8 bits) |

34.3.10.6 CRC

Computation remains unchanged from the 32-bit CRC defined for the 4 Megabits per second data rate. The content of the CRC field is first scrambled with the scheme described later and then encoded with HHH(1, 13) as described previously. Note that the 32 CRC bits for the IrLAP frame are calculated before the IrLAP frame is scrambled. The transmitted CRC field is a 48-chip (2 µs) sequence.

34.3.10.7 FLUSH BYTE (FB)

The Flush Byte (FB) is the 8-bit sequence:

FB = '00'00'00'00'.

These 8 bits are not scrambled but directly sent to the HHH(1,13) encoder. The transmitted FB field is a 12-chip (0.5 µ s) sequence. Note that the FB field is required to enable complete decoding of the CRC field. The flush byte denotes the end of the main body. Since the flush byte is not scrambled, a well-balanced HHH(1,13) sequence precedes the STOP delimiter. This sequence would be equivalent to the UART "Break" command.

34.3.10.8 STOP (STO)

The transmitted STOP (STO) delimiter is the 48-chip (2 µs) sequence

STO = '001'001'010'101'001'000'100'000'100'101'010'100'100'000'100'000'.

The left-most/right-most chip of STO is transmitted first/last and a '1' in STO means an active chip (pulse) and a '0' means an empty chip (no pulse). As in the Start Flag delimiter, the Stop flag also contains a subsequence '1001010101001' that violates the HHH (1,13) code. This subsequence also occurs twice in the Stop Flag delimiter.

34.3.10.9 NULL Sequence

The transmitted NULL sequence is the 24-chip (1 µs) sequence

NULL = '000'000'000'000'000'000'000'000'.

The NULL field is a new field for the purpose of providing an HHH(1,13) code pattern violation that permits terminating reception of the packet in the event that the STO field is not recognized. The left-most/right-most chip in NULL is transmitted first/last and all chips of NULL are empty chips (no pulses). The NULL field increases the probability that the packet is terminated close to the STOP flag. The NULL field also reduces the probability that two back to back packets are interpreted as a single packet, should the STOP flag delimiter of the first packet be missed.

34.3.10.10 Scrambling and Descrambling Functions

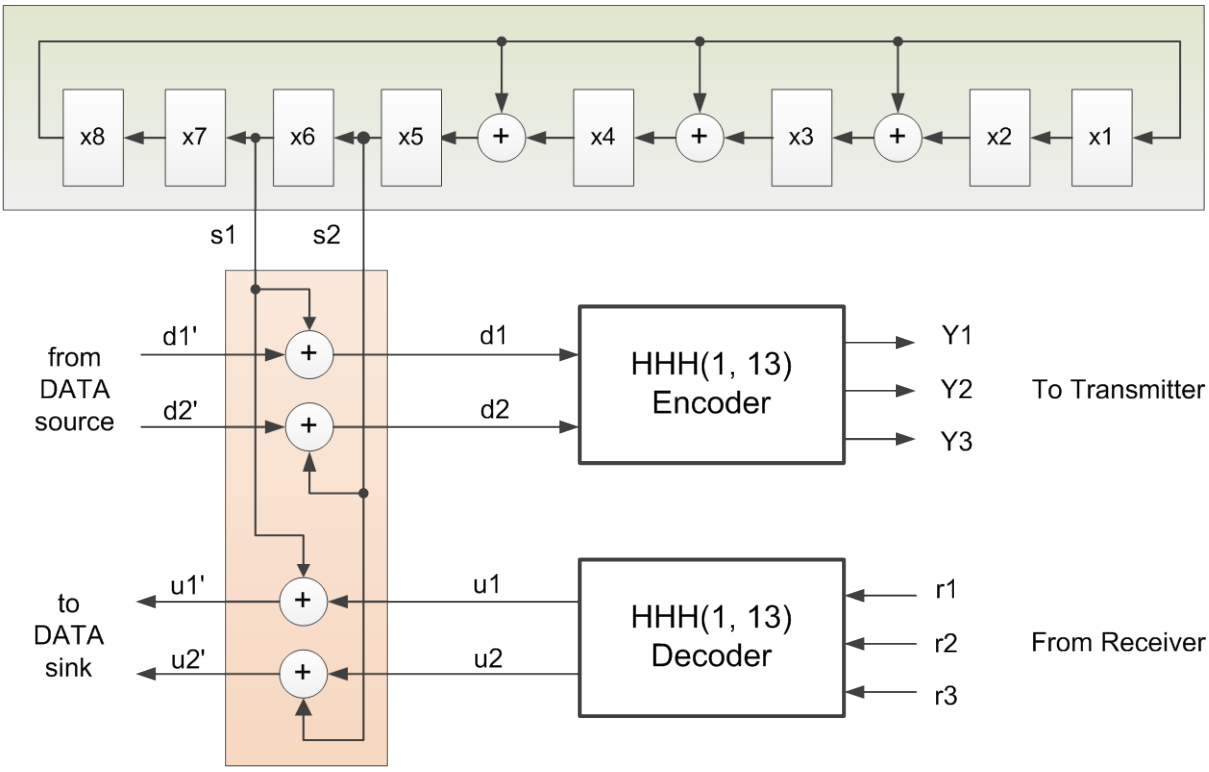
It is advantageous to enhance the encoder/decoder system with simple scrambler/descrambler functions.

The primitive polynomial: $x^8 \oplus x^4 \oplus x^3 \oplus x^2 \oplus 1$,

where $\oplus$ indicates a modulo-2 addition or, equivalently, a logic exclusive OR (XOR) operation. The operations of the proposed scrambling and descrambling functions are performed according to the principles of frame synchronized scrambling/descrambling (FSS) mechanisms. The scrambling/descrambling scheme is shown in the figure below. The linear feedback shift register (LFSR) produces a maximum-length pseudo-random sequence with period 255. The output of register cell x6 shown in the figure is defined to be the equivalent serial output of the LFSR.

The modulo-2 adders shown in the figure below correspond to logic XOR (exclusive OR) gates. During transmission, each new pair of source bits (d1', d2') is XOR-ed with a new pair of scrambling bits (s1, s2) to produce the scrambled data bit pair (d1, d2) entering the encoder. Similarly, during reception, each new pair of decoded bits (u1, u2) is XOR-ed with a new pair of descrambling bits (s1, s2) to produce the descrambled user bit pair (u1', u2') that is sent to the data sink. A scrambling/descrambling cycle has duration 3T seconds where T = 41.7 ns is the chip period.

Figure 131: Scrambling / Descrambling Scheme



34.3.10.11 Effects and Limits of Scrambling/Descrambling

By enhancing the system with scrambling/descrambling functions during data transmission/reception, one achieves generally better duty cycle statistics in the HHH(1,13) coded channel chip stream; the resulting duty cycle converges towards the average duty cycle of the code (≈ 26%) for typical payload data. Scrambling can greatly reduce the probability of occurrence of worst-case patterns.

34.3.10.12 Aborted Packets

Receivers may only accept packets that have valid STA, IrLAP frame, CRC, and STO fields as defined in the Packet Format section. The PA field need not be valid in the received packet. All other packets are aborted and ignored.

34.3.10.13 Back to Back Packet Transmission

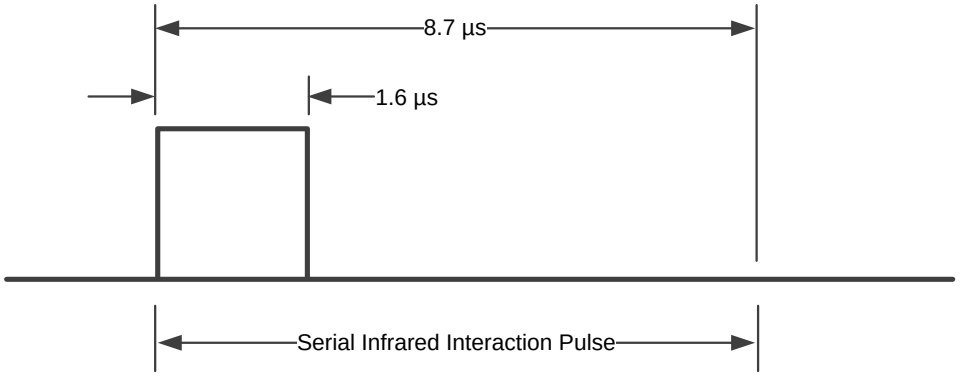
Back to back, or "brick-walled" packets are allowed, but each packet must be complete (i.e., containing PA, STA, IrLAP Frame, CRC and STO fields).

34.3.11 SIR - Serial Interaction Pulse

In order to guarantee non-disruptive coexistence with slower (up to 115.2 Kilobits per second) systems, once a higher speed (above 115.2 Kilobits per second) connection has been established, the higher speed system must emit a Serial infrared Interaction Pulse (SIP) at least once every 500 ms as long as the connection lasts to quiet slower systems that might interfere with the link. The pulse can be transmitted immediately after a packet has been transmitted. Initiation of this pulse will be performed by software by setting a bit in the VFIR Control register.

The pulse is shown below.

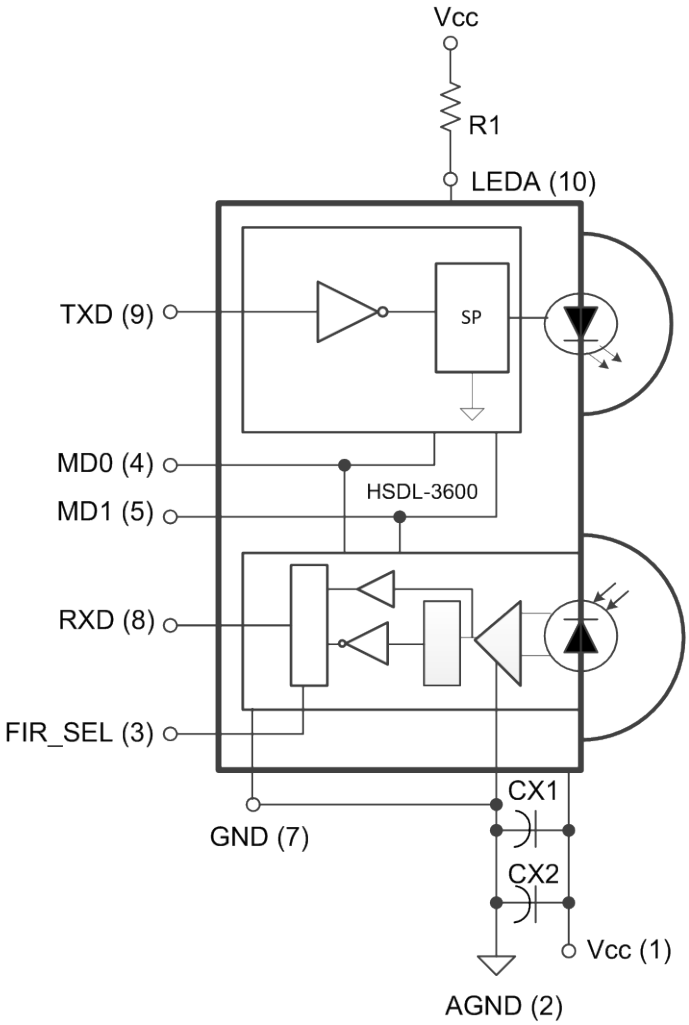
Figure 132: SIP



34.3.11.1 External Interface

The SIR/FIR/VFIR signals are muxed to the external transceiver. The transceiver has TX and RX pins, and may optionally have controls for receiver gain and transmitter power. This module provides the TX and RX pins, and each has a polarity control for compatibility with all vendors. Other transceiver pins are not directly supported, although they could be provided using GPIO signals. Refer to the drawing below for typical interface. TXD and RXD are VFIR module pins. MD and FIR_SEL pins would be GPIO. MD pins are for transmit power. FIR_SEL controls gain of receiver, and should be set to zero for SIR mode.

Figure 133: Typical Interface



34.4 UART Registers

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

The Tegra K1 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.

34.4.1 UART_THR_DLAB_0_0

UART Transmit Holding Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	R/W	Reset	Description
7:0	RO	0x0	THR_A: Transmit holding register, holds the character to be transmitted by the UART. In FIFO mode, a write to this FIFO places the data at the end of the FIFO.
7:0	RO	X	RBR_A: Receive Buffer Register. Rx Data read from here.
7:0	RO	0x0	DLL_A: Divisor Latch LSB (low 8 bits of 16-bit Baud Divisor)

34.4.2 UART_IER_DLAB_0_0

The IER and DLH registers occupy the same address space, selected by the LSR.DLAB bit. The Interrupt Enable Register (IER) is selected if the LSR.DLAB bit is clear. The Divisor Latch MSByte register (DLM) is selected if the LSR.DLAB bit is set. The DLM register holds the upper 8 bits of the 16-bit Baud Divisor (16x).

UART Interrupt Enable por=0x00000000

RX_TIMEOUT occurs when data has been sitting in the Rx FIFO for more than 4 character times without being read because there is not enough data to reach the trigger level. Interrupt needed to handle this case so that all data is received in a timely manner. Note that this normally occurs at the end of an incoming data stream.

EORD (End of Receive Data) Interrupt occurs when the receiver detects that data stops coming in for more than 4 character times. This interrupt is useful for determining that the sending device has completed sending all its data. EORD timeout will not occur if the receiving data stream is stopped because of hardware handshaking.

To clear the EORD timeout interrupt you must DISABLE the EORD interrupt enable (IE_EORD).

Interrupt Enable and Divisor Latch MSByte Registers

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5	0x0	IE_EORD: Interrupt Enable for End of Received Data 0 = DISABLE 1 = ENABLE
4	0x0	IE_RX_TIMEOUT: Interrupt Enable for Rx FIFO timeout 0 = DISABLE 1 = ENABLE
3	0x0	IE_MSI: Interrupt Enable for Modem Status Interrupt 0 = DISABLE 1 = ENABLE
2	0x0	IE_RXS: Interrupt Enable for Receiver Line Status Interrupt 0 = DISABLE 1 = ENABLE
1	0x0	IE_THR: Interrupt Enable for Transmitter Holding Register Empty interrupt 0 = DISABLE 1 = ENABLE
0	0x0	IE_RHR: Interrupt Enable for Received Data Interrupt 0 = DISABLE 1 = ENABLE

34.4.3 UART_IIR_FCR_0

The FCR and IIR registers occupy the same address space. The FIFO Control Register (FCR) is write-only. The Interrupt Identification Register (IIR) is read-only.

UART FIFO Control Register por=0x00000000

The DMA can run in one of two modes:

- If Mode0 is selected (NO_CHANGE), the Rx DMA request goes 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 goes 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 146 Interrupt ID Encoding

IIR[3:0]	Priority Level
0001	No interrupt
0110	Overrun Error, Parity Error, Framing Error, Break
0100	Receiver Data Available
1100	rx_timeout_intr
1000	eord_timeout_intr
0010	Transmitter Holding Register empty
0000	modem_status interrupt

UART FIFO Control and Interrupt Identification Registers

Offset: 0x8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:6	X	EN_FIFO: FIFO Mode Status 0=16450 mode(no FIFO) 1 = 16550 mode (FIFO) 1 = MODE_16550 0 = MODE_16450
7:6	0x0	RX_TRIG: 0 = FIFO_COUNT_GREATER_1 1 = FIFO_COUNT_GREATER_4 2 = FIFO_COUNT_GREATER_8 3 = FIFO_COUNT_GREATER_16

Bit	Reset	Description
5:4	0x0	TX_TRIG: 0 = FIFO_COUNT_GREATER_16 1 = FIFO_COUNT_GREATER_8 2 = FIFO_COUNT_GREATER_4 3 = FIFO_COUNT_GREATER_1
3	X	IS_PRI2: Encoded Interrupt ID Refer to IIR[3:0] table above 0 = DISABLE 1 = ENABLE
3	0x0	DMA: 0:DMA_MODE_0 1:DMA_MODE_1 0 = NO_CHANGE 1 = CHANGE
2	X	IS_PRI1: Encoded Interrupt ID Refer to IIR[3:0] table above 0 = DISABLE 1 = ENABLE
2	0x0	TX_CLR: 1 = Clears the contents of the transmit FIFO and resets its counter logic to 0 (the transmit shift register is not cleared or altered). This bit returns to 0 after clearing the FIFOs. 0 = NO_CLEAR 1 = CLEAR
1	X	IS_PRI0: Encoded Interrupt ID Refer to IIR[3:0] table above 0 = DISABLE 1 = ENABLE
1	0x0	RX_CLR: 1 = Clears the contents of the receive FIFO and resets its counter logic to 0 (the receive shift register is not cleared or altered). This bit returns to 0 after clearing the FIFOs. 0 = NO_CLEAR 1 = CLEAR
0	X	IS_STA: Interrupt Pending if ZERO 0 = INTR_PEND 1 = NO_INTR_PEND
0	0x0	FCR_EN_FIFO: 1 = Enable the transmit and receive FIFOs. This bit should be enabled 0 = DISABLE 1 = ENABLE

34.4.4 UART_LCR_0

UART Line Control Register

Offset: 0xc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	DLAB: Divisor Latch Access Bit (set to allow programming of the DLH, DLM Divisors) 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	0x0	SET_B: Set BREAK condition -- Transmitter sends 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 1 = WORD_LENGTH_6 2 = WORD_LENGTH_7 3 = WORD_LENGTH_8

34.4.5 UART_MCR_0

The RTS and CTS pins are used for hardware handshaking with an external serial device. RTS (Request-To-Send) informs the device that the UART is ready to accept data. CTS (Clear-To-Send) comes from the RTS of the external device and indicates that data can be sent.

The RTS pin can be controlled manually with the RTS bit in the MCR or automatically by setting the RTS_EN bit.

If RTS_EN is set, the RTS pin automatically removes the Request-To-Send when the FIFO reaches the trigger level.

If the CTS_EN bit is set, the UART transmitter stops transmitting when the Clear-To-Send input is taken away.

UART Modem Control Register

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6	0x0	RTS_EN: 1 = Enable RTS Hardware Flow Control 0 = DISABLE 1 = ENABLE
5	0x0	CTS_EN: 1 = Enable CTS Hardware Flow Control 0 = DISABLE 1 = ENABLE
4	0x0	LOOPBK: 1 = Enable internal loop back of Serial Out to In 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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 hardware flow control not enabled 0 = FORCE_RTS_HI 1 = FORCE_RTS_LOW
0	0x0	DTR: 1 = Force DTR to high 0 = FORCE_DTR_HI 1 = FORCE_DTR_LOW

34.4.6 UART_LSR_0

UART Line Status Register

Offset: 0x14 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8	X	TX_FIFO_FULL: Transmitter FIFO full status 0 = NOT_FULL 1 = FULL
7	X	FIFOE: 1 = Receive FIFO Error 0 = NO_ERR 1 = ERR
6	X	TMTY: Transmit Shift Register empty status 0 = NO_EMPTY 1 = EMPTY
5	X	THRE: 1 = Transmit Holding Register is Empty -- OK to write data 0 = FULL 1 = EMPTY
4	X	BRK: 1 = BREAK condition detected on line 0 = NO_BREAK 1 = BREAK
3	X	FERR: 1 = Framing Error 0 = NO_FRAME_ERR 1 = FRAME_ERR
2	X	PERR: 1 = Parity Error 0 = NO_PARITY_ERR 1 = PARITY_ERR
1	X	OVRF: 1 = Receiver Overrun Error 0 = NO_OVERRUN_ERROR 1 = OVERRUN_ERROR
0	X	RDR: 1 = Receiver Data Ready (Data available to read) 0 = NO_DATA_IN_FIFO 1 = DATA_IN_FIFO

34.4.7 UART_MSR_0

UART Modem Status Register

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	CD: State of Carrier detect pin 0 = DISABLE 1 = ENABLE
6	0x0	RI: State of Ring Indicator pin 0 = DISABLE 1 = ENABLE
5	0x0	DSR: State of Data set ready pin 0 = DISABLE 1 = ENABLE
4	0x0	CTS: State of Clear to send pin 0 = DISABLE 1 = ENABLE
3	0x0	DCD: Change (Delta) in CD state detected 0 = DISABLE 1 = ENABLE
2	0x0	DRI: Change (Delta) in RI state detected 0 = DISABLE 1 = ENABLE
1	0x0	DDSR: Change (Delta) in DSR state detected 0 = DISABLE 1 = ENABLE
0	0x0	DCTS: Change (Delta) in CTS state detected 0 = DISABLE 1 = ENABLE

34.4.8 UART_SPR_0

UART Scratchpad Register

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	SPR_A: Scratchpad register (not used internally)

34.4.9 UART_IRDA_CSR_0

Bits [7:6] control the infrared (SIR) encoding. If enabled, each zero bit is transmitted as a short IR pulse. No pulse is sent during idle or '1' data bits. The IR pulse can be either 3/16 or 4/16 of a normal bit time.

The low 4 bits control signal polarity and apply whether or not SIR coding is enabled.

UART IrDA Pulse Coding CSR Register

Offset: 0x20 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00xx0000)

Bit	Reset	Description
7	0x0	SIR_A: 1 = Enable SIR coder 0 = Disable SIR coder 0 = DISABLE 1 = ENABLE
6	0x0	PWT_A: 0=3/16th Baud Pulse, 1=4/16 0 = BAUD_PULSE_3_14 1 = BAUD_PULSE_4_14
3	0x0	INVERT_RTS: Inverts the normally inactive high nRTS pin 0 = DISABLE 1 = ENABLE
2	0x0	INVERT_CTS: Inverts the normally inactive high nCTS pin 0 = DISABLE 1 = ENABLE
1	0x0	INVERT_TXD: Inverts the normally inactive high TXD pin 0 = DISABLE 1 = ENABLE
0	0x0	INVERT_RXD: Inverts the normally inactive high RXD pin 0 = DISABLE 1 = ENABLE

34.4.10 UART_RX_FIFO_CFG_0

Offset: 0x24 │ Read/Write: R/W │ Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x000001)

Bit	Reset	Description
7	0x0	EN_RX_FIFO_TRIG: Enable use of RX_FIFO_TRIG count (obsoletes RX_TRIG when enabled) 0 = DISABLE 1 = ENABLE
5:0	0x1	RX_FIFO_TRIG: Set RX_FIFO trigger level (any value 1 thru 32)

34.4.11 UART_MIE_0

UART Modem Interrupt Enable Register

Offset: 0x28 │ Read/Write: R/W │ Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111)

Bit	Reset	Description
3	0x1	DCD_INT_EN: Interrupt Enable for Change (Delta) in CD state detected 0 = DISABLE 1 = ENABLE
2	0x1	DRI_INT_EN: Interrupt Enable for Change (Delta) in RI state detected 0 = DISABLE 1 = ENABLE
1	0x1	DDSR_INT_EN: Interrupt Enable for Change (Delta) in DSR state detected 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	0x1	DCTS_INT_EN: Interrupt Enable for Change (Delta) in CTS state detected 0 = DISABLE 1 = ENABLE

34.4.12 UART_ASR_0

The UART has autobaud sensing logic that can measure the width of the start bit of incoming data to determine baud rate. For this to work, the incoming character must have its LSB set. A <cr> works well (0x0d). The logic uses the UART device clock to count how wide the start pulse is, and the BUSY bit is set when the sensing is complete. The value in {RX_RATE_SENSE_H,RX_RATE_SENSE_L} should be divided by 16 with Round-to-Nearest, and the resulting value loaded into the DLM,DLL register pair to set the Baud Clock to 16X the Baud Rate.

UART Auto Sense Baud Register

Offset: 0x3c │ Read/Write: R/W │ Reset: 0x00000000 (0b00xxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	VALID: This bit is set when the controller finishes counting the clocks between two successive clock edges after there is a write to ASR with don't care data. 0 = UN_SET 1 = SET
30	0x0	BUSY: This bit is set when there is a write to ASR and is reset when the controller finishes counting the clock edges between two successive clock edges. 0 = NO_BUSY 1 = BUSY
15:8	0x0	RX_RATE_SENSE_H: Shows bits [15:8] of the count of clock edges between two successive clock edges.
7:0	0x0	RX_RATE_SENSE_L: Shows bits [7:0] of the count of clock edges between two successive clock edges.

34.5 VFIR Registers

34.5.1 VFIR_CTL_0

The VFIR Control register is used to:

- enable/disable entire IRDA module
- configure the controller in FIR mode or VFIR mode or SIR or UART mode
- configure either in Transmit mode or Receive mode
- invert the polarity of Tx or Rx line
- invert the polarity of Tx or Rx line
- enable/disable DMA
- know the attention levels of Transmit FIFO and Receive FIFO
- start a transmit or receive operation
- select the frequency
- clear the Transmitter FIFO/Receiver FIFO

VFIR Control Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000020 (0b000xxxx0x000x000xx00000000100000)

Bit	Reset	Description
31	0x0	GLOBAL_ENABLE: 0 = Entire IRDA module is disabled 0 = DISABLE 1 = ENABLE
30	0x0	GO: Set to 1 to start transmit or receive operation, self-clearing 0 = STOP 1 = START
29	0x0	AUTO_RESTART: Set to 1 to restart another operation when previous is done 0 = STOP 1 = RE_START
24	0x0	FORCE_BAD_CRC: Debugging bit to force a CRC error 0 = DISABLE 1 = ENABLE
22	0x0	TX_FIFO_CLR: Clear the Transmitter FIFO 0 = DISABLE 1 = ENABLE
21:20	0x0	TX_ATN_LVL: 00: when not full 3 = slots_empty_12 2 = slots_empty_8 1 = slots_empty_4 0 = not_full
18	0x0	RX_FIFO_CLR: Clear the Receiver FIFO 0 = DISABLE 1 = ENABLE
17:16	0x0	RX_ATN_LVL: 00: when not empty 3 = slots_full_12 2 = slots_full_8 1 = slots_full_4 0 = not_empty
13	0x0	START_SIP: Self-clearing bit to initiate a Serial Ir Interaction Pulse.
12	0x0	EN_PULSECORR: Fixes single pulse errors caused by VFIR propagation effects. 0 = DISABLE 1 = ENABLE
11	0x0	DMA_EN: DMA Enable Allow requests to go to the APB_DMA controller. 0 = DISABLE 1 = ENABLE
10	0x0	TX_TERM: of outgoing frame, i.e., the calculated CRC will be sent, followed by the frame stop sequence. The interrupt will be generated if enabled. 0 = ABORT 1 = NORMAL
9	0x0	COUNT_ODATA: When enabled, the controller will end the frame when the Outgoing Frame Data Length (OFDL) count is reached. Use in conjunction with TX_Term above so controller will know when and how to end the frame. 0 = DISABLE 1 = ENABLE
8	0x0	FORCEBRK: break state (zero) regardless of what the transmitter is doing. 0 = TX_ENABLE 1 = TX_DISABLE

Bit	Reset	Description
7	0x0	NEGATE_RX: Invert polarity on incoming RX pin (applies to all modes) 0 = DISABLE 1 = ENABLE
6	0x0	NEGATE_TX: Invert polarity on outgoing TX pin (applies to all modes) 0 = DISABLE 1 = ENABLE
5:4	0x2	FREQUENCY: The 8 and 24 MHz clocks can be used to save power during transmits. The 72 MHz clock can be used in case the 48 MHz clock is not available due to system constraints. 0 = F8MHz 1 = F24MHz 2 = F48MHz 3= F72MHz
3	0x0	TRANSMIT: 0 = Receive (FIR/VFIR modes) 1 = TRANSMIT 0 = RECEIVE
2:0	0x0	MODE: 000: UART uses the UART B module 7 = RSVD 6 = RSVD1 5 = VFIR 4 = FIR 3 = RSVD2 2 = MIR 1 = SIR 0 = UARTB

34.5.2 VFIR_STS_0

The VFIR Status and Interrupt Identification register is used to get the following status:

- Transmitter/Receiver FIFO Trigger level status
- Transmitter/Receiver FIFO FULL/EMPTY status
- Transmitter/Receiver FIFO empty slots
- whether a Transmit or Receive operation is done or not
- whether End of Frame is received or not
- whether there is any CRC error in received data or not
- whether there is any error in the received data
- whether the controller is busy with transmitting/receiving the data
- whether the VFIR interrupt is generated or not. To generate the interrupt, the corresponding enable bit has to be set in the VFIR Interrupt Enable Register

VFIR Status and Interrupt Identification Register

Offset: 0x4 │ Read/Write: R/W │ Reset: 0xXXXXX0X0 (0bxxxxxxxxxxxxxxxxxxxx0xxx0x000000)

Bit	R/W	Reset	Description
31	RO	X	TX_FTRIG: Transmitter FIFO Trigger level Reached
30	RO	X	TX_FIFO_FULL: Transmitter FIFO is Full
29	RO	X	TX_FIFO_EMPTY: Transmitter FIFO is Empty
28:24	RO	X	TX_FIFO_EMPTY_COUNT: Number of empty slots (words) in the Tx FIFO. The FIFO is a shared Rx/Tx FIFO with a total of decimal 20 entries

Bit	R/W	Reset	Description
23	RO	X	RX_FTRIG: Receiver FIFO Trigger level Reached
22	RO	X	RX_FIFO_FULL: Receiver FIFO is Full
21	RO	X	RX_FIFO_EMPTY: Receiver FIFO is Empty
20:16	RO	X	RX_FIFO_FULL_COUNT: Number of words available to read in the Rx FIFO. The FIFO is a shared Rx/Tx FIFO with a total of decimal 20 entries
15	RO	X	INTR: Global VFIR interrupt (OR of all FIR/VFIR interrupts)
14	RO	X	BUSY: Transmitting or receiving data. It is clear when the controller is idle or transmits a Serial infrared Interaction Pulse (SIP)
11	RW	0x0	TX_UNDRN: break. This bit is cleared upon reading from the register.
7	RW	0x0	MISSED_PACKET: Another Rx arrived before the previous was serviced. Serviced is defined as clearing the Rx_EOF flag. This bit is cleared by writing a 1 to this bit.
6	RO	X	BITSYNC_LOCK: Debug status bit which indicates that receiver is Locked to incoming bitstream.
5	RW	0x0	RX_DETECT: Activity detected on Rx pin This bit is cleared by writing a 1 to this bit.
4	RW	0x0	RX_ERR: Receiver Error. This bit is set when an unexpected or illegal data has been received. This can be a break in transmission, illegal chip values or framing errors. This bit is cleared by writing a 1 to this bit.
3	RW	0x0	RX_FOVRN: Receiver FIFO overrun error occurred. This bit is cleared by writing a 1 to this bit.
2	RW	0x0	RX_CRC_ERR: CRC check on input data failed. This bit is cleared by writing a 1 to this bit.
1	RW	0x0	RX_EOF: Received End of Frame. Set when the last byte in frame is within the receiver FIFO. This bit is cleared by writing a 1 to this bit.
0	RW	0x0	DONE: Done flag. Set when controller completes a Transmit or Receive packet, even if the packet finished early due to errors. This bit is cleared by writing a 1 to this bit.

34.5.3 VFIR_IER_0

VFIR Interrupt Enable Register is used to generate a 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 an interrupt is generated. Similar is the case for other bits in this register

VFIR Interrupt Enable Register

Offset: 0x8 │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxx0xxxxxxxxxxx0xxx0x000000)

Bit	Reset	Description
31	0x0	IE_TX_FTRIG: Enable Interrupt on Transmitter FIFO Trigger level reached 0 = DISABLE 1 = ENABLE
23	0x0	IE_RX_FTRIG: Enable Interrupt on Receiver FIFO Trigger level reached 0 = DISABLE 1 = ENABLE
11	0x0	IE_TX_UNDRN: Transmit a break for the receiving side to abort reception. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	0x0	IE_MISSED_PACKET: Enable Interrupt for Missed Packet error 0 = DISABLE 1 = ENABLE
5	0x0	IE_RX_DETECT: Enable Interrupt on Activity on Rx Pin 0 = DISABLE 1 = ENABLE
4	0x0	IE_RX_ERR: Enable Interrupt on Receiver Error 0 = DISABLE 1 = ENABLE
3	0x0	IE_RX_FOVRN: Enable Interrupt on Receiver FIFO overrun 0 = DISABLE 1 = ENABLE
2	0x0	IE_RX_CRC: Enable Interrupt for input CRC check failure 0 = DISABLE 1 = ENABLE
1	0x0	IE_RX_EOF: Enable Interrupt for Received End of Frame 0 = DISABLE 1 = ENABLE
0	0x0	IE_DONE: Enable Interrupt for Done Status 0 = DISABLE 1 = ENABLE

34.5.4 VFIR_OFDL_0

VFIR Outgoing Frame Data Length is used to configure the number of bytes of data to be sent.

VFIR Outgoing Frame Data Length

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

Bit	Reset	Description
15:0	0x0	OFDL: Set to the length in bytes of the outgoing frame. Length is calculated on the information field only (address, control and data fields only) without the CRC and flags. When in this mode, the amount of data transferred to the controller is counted and when the count is reached the controller finishes sending the frame in normal fashion by transmitting CRC field, STO flag and then the SIP.

34.5.5 VFIR_IFDL_0

VFIR Incoming Frame Data Length gives the information of the data received in bytes.

VFIR Incoming Frame Data Length

Offset: 0x10 │ Read/Write: RO │ Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	IFDL: Running length in bytes of the incoming data stream. At the end of frame reception when the Received End of Frame interrupt occurs, this register reflects the length of the information field received in that frame, not counting the CRC field and other flags. Resets on the start of next frame reception, when a new STA field has been detected.

34.5.6 VFIR_FIFO_0

The VFIR FIFO is a shared FIFO. It holds the data in Transmitter/Receiver mode.

VFIR.FIFO (VFIR FIFO)

Offset: 0x40 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA: Shared FIFO. When transmitting, this FIFO holds the data to be transmitted. When receiving, this FIFO holds the received data.

35.0 SERIAL PERIPHERAL INTERFACE (SPI) CONTROLLER

The Serial Peripheral Interface (SPI) controller is a serial communications link between the processor and on-chip/off-chip serial peripheral devices such as sensors, ADC/DAC devices, and Flash controllers. Tegra® K1 devices support both Master and Slave modes of SPI operation on this interface.

In this section:

- *Word* refers to 32-bit data in the TX or RX FIFO
- *Packet* refers to variable length data that is transmitted or received
  - In Unpacked Mode, packet size can be any length from 4 to 32 bits in Master mode or 8 to 32 bits in Slave mode
  - In Packed Mode, packet size can be 4, 8, 16, or 32 bits

35.1 Functionality

The Tegra K1 SPI Controller works as a Master/Slave on the SPI bus. It has independent transmit and receive FIFOs of 64 x 32 bits each. Software can program the controller to generate transactions of a required packet length on the SPI bus, where a transaction is a sequence of packets in either direction.

The controller supports either APB DMA or program transfer to read and write from the FIFOs as required. At the end of each transaction, an interrupt can be generated, if enabled. Software uses transmit and receive operations in combination with chip select (CS) control to generate commands on the SPI bus.

The interface consists of four signals: Chip Select/Slave Select (CS_N), Clock (SCLK), Master Data Out and Slave Data In (MOSI), and Master Data In and Slave Data Out (MISO).

Features

- Master and Slave functionality
  - Master: All transmission format modes are supported for both transmit and receive operations
  - Slave (transmits): Modes 1 and 3 are supported
  - Slave (receives): All modes are supported
- Independent Rx and Tx FIFOs
- FIFO depth of 64 x 32 bits
- Programmable bit lengths from 3 to 31 bits, resulting in packet sizes of 4 to 32 bits
- PIO or DMA Mode depending on packet size

  Options include Packed/Unpacked, Full-Duplex Mode, Both Enable Bit (x2 Mode), Both Enable Byte (x2 Mode), Bidirectional, Least Significant Bit, Least Significant Byte First (Endian-ness), software or hardware chip-select polarity selection
- Programmable clock phase and polarity
- Programmable delay between consecutive transfers
- Packed mode support for bit lengths of 3 (4-bit packet size), 7 (8-bit packet size), and 15 (16-bit packet size)
- Chip select (CS) can be controlled by software or can be generated automatically by hardware on packet boundaries
- Maximum 4-chip support with programmable CS polarity for each chip select
- DMA support
- Simultaneous receive and transmit operations supported
- The maximum frequency of the device clock is 50 MHz, so the maximum data rate is 50 Mbits/s

35.1.1 Transmission Format

Using the Mode field in the SPI Command1 Register, software sets one of the four modes in which the SPI controller works. The two bits of the Mode field specify the idle (inactive) state of SCLK before the chip select is asserted and the data transmitting/receiving edges of SCLK. Mode [1] determines the SCLK polarity. If the polarity is 0, then the SCLK signal is 0 when idle and transitions to 1 when data transfer starts. If the clock polarity is 1, then the SCLK signal is 1 when idle and transitions to 0 when data transfer starts. Mode [0] is the SCLK phase control bit. If the clock phase is 0, then the receiver latches the data on the first transition of SCLK from the idle state. If the clock phase is 1, then the receiver latches the data on the second transition of SCLK.

Modes supported by the SPI controller are:

- Master: All transmission format modes are supported both for transmitting and receiving
- Slave: Modes 1 and 3 are supported for transmitting
- Slave: All modes are supported for receiving

The following table defines the four modes for the SPI protocol.

Table 147: SPI Modes with Clock Polarity and Phase

SPI Mode	Clock Polarity	Clock Phase	SCLK Inactive State	Data Latch IN	Data Latch OUT
0	0	0	Low	Latched IN on the positive edge of clock	Latched OUT on the negative edge of clock
1	0	1	Low	Latched IN on the negative edge of clock	Latched OUT on the positive edge of clock
2	1	0	High	Latched IN on the negative edge of clock	Latched OUT on the positive edge of clock
3	1	1	High	Latched IN on the positive edge of clock	Latched OUT on the negative edge of clock

35.1.2 Modes of Operation

35.1.2.1 Master and Slave Modes

The SPI controller can be configured to operate as a Master or Slave. When the M/S bit in the SPI Command1 Register is set, Master mode is selected; when the M/S bit in the SPI Command1 Register is cleared, Slave mode is selected. Only the SPI Master can initiate a transaction.

In Master mode, data from the Tx FIFO is transmitted on the MOSI pin. The data from the slave is received on the MISO pin and sent to the Rx FIFO. The Master can simultaneously transmit and receive on MOSI and MISO in Full-Duplex Mode.

In Slave mode, once software enables the SPI controller by setting the PIO bit, it simply waits for the Master to initiate a transaction. Before the transaction begins, the slave logic continuously polls the CS input. When the Master asserts CS and SCLK is transmitted to the Slave, the Slave data is transmitted from the Tx FIFO on MISO and data from MOSI is received in the Rx FIFO. The Slave can also simultaneously transmit and receive on MOSI and MISO in Full-Duplex Mode.

Table 148: Master Mode Port Configuration

Name	Direction	Description
MOSI	Bidirectional	Output in Full-Duplex/Tx Mode Input in Both_Enable Rx Mode
MISO	Bidirectional	Input in Full-Duplex/Rx Mode
SCLK	Output	Output in Both_Enable TX Mode
CS_#x	Output	Slave select signal to x, where x is a number between 0 to 3

Table 149: Slave Mode Port Configuration

Name	Direction	Description
MISO	Bidirectional	Output in Full-Duplex/Tx Mode Input in Both_Enable Rx Mode
MOSI	Bidirectional	Input in Full-Duplex/Rx Mode Output in Both_Enable TX Mode
SCLK	Input	Synchronization clock received from the Master
CS_#	Input	Select signal

When the SPI controller is configured to interface with multiple slaves, the controller has one CS signal for each slave, up to a maximum of four slaves. During a transfer, the master asserts CS specified in the CS_SEL bits in SPI Command1 Register. There can be no more than one slave transmitting data during any particular transfer.

Known Limitations

In Slave Mode, the maximum block size supported by the SPI controller is 0xFFFB with packed mode and bit lengths of 8 or 16.

35.1.2.2 DMA and PIO Modes

PIO and DMA are the two main operational modes for the SPI controller. PIO mode is used when the number of packets to the transmitter or receiver is less than or equal to 64. DMA mode is used when the number of packets to the transmitter or receiver is greater than 64. In PIO or DMA mode, the SPI controller sends an interrupt to the processor at the end of each transfer. In this mode software configures the number of packets and number of bits in each packet to be transmitted/received. The limitation of PIO Mode is that it can be used only if the maximum number of packets that can be transferred is less than or equal to 64 packets (the FIFO depth is 64). If software wants to transmit/receive more than 64 packets in PIO Mode, it has to reconfigure the SPI controller every 64 packets.

To send/receive more than 64 packets, the SPI Controller can be configured in DMA mode (enabled by writing 1 to the DMA_EN bit in the SPI DMA Control Register). The SPI controller transmits or receives the number of packets as indicated by the BLOCK_SIZE field in the SPI Block Size Register. BLOCK_SIZE is a 16-bit field, so 2^{16} packets (amounting to 256 KB) can be transmitted/received in a single software configuration.

CS usage guideline: If a hardware-based CS is used, the CS goes to the inactive state after the transfer of packets as programmed in the BLOCK_SIZE register field is completed. If CS has to stay active in multiple consecutive transactions (for

example, the first transaction – Tx: 64 Packets; the second transaction – Rx: 128 Packets), then the software based CS has to be used (in hardware-based CS, CS goes to the inactive state after the completion of the first transaction).

35.1.2.3 Packed and Unpacked Modes

SPI communication can be carried out in Packed or Unpacked Mode. These are both present in DMA and PIO Modes.

- Packed Mode: In Packed Mode, data can be transmitted/received with each packet size equal to 4, 8, 16, or 32 bits. The PACKED field and BIT_LEN field in the DMA Command1 Register determine if the mode Packed or Unpacked and the length of each packet. If the BIT_LEN field is set to 03h, each packet is 4 bits long and each word in the Tx/Rx FIFO contains 8 such packets. Similarly, if BIT_LEN is 07h, the packet length is 8. If BIT_LEN is 0Fh, the packet length is 16. If BIT_LEN is 1Fh, the packet length is 32. In Packed Mode, if BIT_LEN is set to 03h and if the number of packets is set to a non-multiple of 4 (for example, 5), the last word in FIFO contains only the fifth packet with 4 valid bits; 28 extra invalid bits will be ignored by the controller for transmits and will contain invalid data for receives.
- Unpacked Mode: If BIT_LEN is set to 03h, then each word in the Tx/Rx FIFO has 4 bits with the remaining bits in each word being ignored. BIT_LEN can be any value from 3 to 31 in Unpacked Mode.

35.1.2.4 Bidirectional Mode

Bidirectional mode is selected when the BIDIR bit is set in the SPI Command1 Register. In this mode, the SPI controller uses only one serial data pin for the interface with external device(s). The M/S bit decides which pin to use. The MOSI pin becomes the serial data I/O (MOMI) pin for the Master Mode, and the MISO pin becomes serial data I/O (SISO) pin for the Slave Mode. The SPI controller does not use the MISO pin in Master Mode and the MOSI pin in Slave Mode.

Table 150: Bidirectional Mode Port Configuration for Master (M/S Bit Set)

Name	Direction	Description
MOSI > MOMI	Bidirectional	Data Output and Input to Slave(s)
MISO > --	--	--
SCLK	Output	Synchronization clock to all Slaves
CS	Output	Slave select signal to x, where x is a number from 0 to 3, inclusive

Table 151: Bidirectional Mode Port Configuration for Slave (M/S Bit Cleared)

Name	Direction	Description
MOSI > --	--	--
MISO > SISO	Bidirectional	Data Output and Input from Slave(s)
SCLK	Input	Synchronization clock
CS	Input	Select signal

35.1.2.5 Both Enable Byte and Bit Modes (SPI x2 Mode)

The Both Enable Byte Mode is selected when the BOTH_EN_BYTE bit is set in the SPI Command1 Register. In this mode, the SPI controller uses both data pins only for transmitting or only for receiving. When the BOTH_EN_BYTE register field is set to 1, the Tx and Rx bits decide whether to transmit or receive. When the Tx bit is set, the MOSI and MISO pins are used to transmit serial data from the Tx FIFO and Rx FIFO at the same time. When the Rx bit is set, the MISO and MOSI pins are used to receive serial data onto Tx FIFO and Rx FIFO at the same time.

Table 152: Both Enable Mode Port Configuration for Master (Tx Bit Set)

Name	Direction	Description
MOSI	Output	Data Output to Slave(s)
MISO	Output	Data Output to Slave(s)
SCLK	Output	Synchronization clock to all Slaves
CS_N_OUT	Output	Slave select signal to x, where x is a number from 0 to 3, inclusive

Table 153: Both Enable Mode Port Configuration for Slave (Rx Bit Set)

Name	Direction	Description
MOSI	Input	Data Input from Slave(s)
MISO	Input	Data Input from Slave(s)
SCLK	Input	Synchronization clock
CS_N	Input	Select signal

The Both Enable Bit Mode is selected when the BOTH_EN_BIT bit is set in the SPI Command1 Register. This feature is different from Both Enable Byte as follows: In Both Enable Byte Mode, a packet from the Tx FIFO is transmitted or received on the MOSI data line and a packet from the Rx FIFO is transmitted or received on the MISO data line. In Both Enable Bit Mode, a packet from the Tx FIFO is transmitted or received on both MOSI and MISO data lines. Even bits are transmitted or received on the MOSI data line and odd bits are transmitted or received on the MISO data line. This is the same in both Master and Slave Modes.

> **Note**: BOTH_EN_BYTE and BOTH_EN_BIT should not be set to 1 at the same time. BOTH_EN_BYTE or BOTH_EN_BIT should not be set to 1 when the BIDIR bit is set to 1.

35.1.2.6 En_LE_Bit and En_LE_Byte Modes

> **Note**: En_LE_Bit is Enable Little Endian Bit, and En_LE_Byte is Enable Little Endian Byte.

These two bits allow a data packet to be transmitted or received in Little Endian or Big Endian format. Once a data packet is read from the Tx FIFO, it undergoes a two-step pre-processing based on En_LE_Byte and En_LE_Bit.

> Step 1. In this step, En_LE_Byte configures whether a packet has to be arranged internally according to the Little Endian byte format or Big Endian byte format. For example, if a 2-byte packet is written into the FIFO by software as Byte 1,Byte 0, setting En_LE_Byte = 0 makes the SPI controller arrange the packet internally as Byte 0,Byte 1(Big

Endian format). If En_LE_Byte = 1, the SPI controller internally arranges the packet as Byte 1,Byte 0 (Little Endian format) before the packet is transmitted.

Step 2. In this step, the output of Step 1 is taken, and data is put on the MOSI/MISO line according to En_LE_Bit. If En_LE_Bit is set to 0, the most significant bit of the output of Step 1 is put on the MOSI/MISO line first. If En_LE_Bit is set to 1, the least significant bit of Step 1 output is put on the MOSI/MISO line.

These steps are illustrated in the following table.

Step 1	**FIFO Packet (MSB - LSB)**	**Byte**	**Output of Step 1**
	Byte 1, Byte 0	0	Byte 0, Byte 1
		1	Byte 1, Byte 0
Step 2	**Output of Step 1**	**Bit**	**Data Transmitted on MOSI (Right-most Bit Sent First)**
	Byte 0, Byte 1	0	Byte 1 (0), Byte 1 (1), …, Byte 0 (6), Byte 0 (7)
	Byte 0, Byte 1	1	Byte 0 (7), Byte 0 (6), …, Byte 1 (1), Byte 1 (0)
	Byte 1, Byte 0	0	Byte 0 (0), Byte 0 (1), …, Byte 1 (6), Byte 1 (7)
	Byte 1, Byte 0	1	Byte 1 (7), Byte 1 (6), …, Byte 0 (1), Byte 0 (0)

Similarly for Receive operations, the first packet received will be internally rearranged first based on En_LE_Bit, then the output of the previous step will be rearranged based on En_LE_Byte before writing the final packet into the Rx FIFO.

The following examples explain the effect of these two bits:

- Unpacked Mode
  - Tx: The following example shows how a 20-bit length packet from the FIFO is transmitted. The right-most bit in Column D in the table below is transmitted first. B.7-B.0 indicates that bit 0 is transmitted first followed by bit 1 until bit 7, which is transmitted last.

A	B	C	D
FIFO WORD: 20 (MSB) – 0 (LSB)	**En_LE_Bit**	**En_LE_Byte**	**Data Transmitted on MOSI/MISO Line**
	0	0	16, 17,18, 19, 20, B.8-B.15, B.0-B.7
	0	1	B.0-B.7, B.8-B.15, 16, 17, 18, 19, 20
	1	0	B.7-B.0, B.15-B.8, 20, 19, 18, 17, 16
	1	1	B.20-B.0

  - Rx: The following example shows how a 20-bit length packet from the FIFO is received. The right-most bit in Column D in the table below is received last on the MOSI/MISO lines. B.7-B.0 indicates that bit 7 is received first followed by bit 6 until bit 0, which is received last. B.20 – B.0 are always written into the FIFO after the En_LE_Byte and En_LE_Bit transformations.

A	B	C	D
FIFO WORD: 20 (MSB) – 0 (LSB)	**En_LE_Bit**	**En_LE_Byte**	**Data Received on MOSI/MISO Line**
	0	0	B.7-B.0, B.15-B.8, 20, 19, 18, 17, 16
	0	1	20, 19, 18,17, 16, B.15-B.8, B.7-B.0

A	B	C	D
FIFO WORD: 20 (MSB) – 0 (LSB)	**En_LE_Bit**	**En_LE_Byte**	**Data Received on MOSI/MISO Line**
	1	0	16, 17, 18, 19, 20, B.8-B.15, B.0-B.7
	1	1	B.0-B.20

- Packed Mode
  - Tx: The following example shows how 3 16-bit packets are transmitted. Column B indicates the valid packets to be transmitted. Because FIFO Word2 is not transmitted entirely, valid packets are only from B.15-B.0. In Column D, P1(0 -15) indicates Packet1 of Column B is transmitted, B.15 is transmitted first, and B.0 is transmitted last.

A	B: Valid Packets	B	C	D
FIFO WORD1: 31 (MSB) – 0 (LSB)	**P1(15-0), P0(15-0)**	**En_LE_Bit**	**En_LE_Byte**	**Data Transmitted on MOSI/MISO Line**
FIFO WORD2: 31 (MSB) – 0 (LSB)	**P2(15-0)**	0	0	P2(8-15), P2(0-7), P1(8-15), P1(0-7), P0(8-15), P0(0-7)
		0	1	P2(0-7), P2(8-15), P1(0-7), P1(8-15), P0(0-7), P0(8-15)
		1	0	P2(7-0), P2(15-8), P1(7-0), P1(15-8), P0(7-0), P0(15-8)
		1	1	P2(15-8), P2(7-0), P1(15-8), P1(7-0), P0(15-8), P0(7-0)

  - Rx: The following example shows how 3 16-bit packets are received. Column B indicates the valid received packets in the Rx FIFO. In column D, P1(0 -15) indicates Packet1 of Column B is received, B.0 is received first, and B.15 is received last.

A	B: Valid Packets	B	C	D
FIFO WORD1: 31 (MSB) – 0 (LSB)	**P1(15-0), P0(15-0)**	**En_LE_Bit**	**En_LE_Byte**	**Data Received on MOSI/MISO Line**
FIFO WORD2: 31 (MSB) – 0 (LSB)	**P2(15-0)**	0	0	P0(7-0), P0(15-8), P1(7-0), P1(15-8), P2(7-0), P2(15-8)
		0	1	P0(15-8), P0(7-0), P1(15-8), P1(7-0), P2(15-8), P2(7-0)
		1	0	P0(8-15), P0(0-7), P1(8-15), P1(0-7), P2(8-15), P2(0-7)
		1	1	P0(15-8), P0(7-0), P1(15-8), P1(7-0), P2(15-8), P2(7-0)

35.2 SPI Programming Guidelines

There are two basic modes of operation: DMA and PIO modes. It is required that software sets up all parameters in the SPI Command1 and Command2 registers, SPI CS Timing 1 and 2 registers, SPI FIFO Control/Status register, SPI DMA Control register, and SPI Block Size register before enabling the PIO bit in any of these modes.

35.2.1 DMA Mode

This mode is enabled by writing 1 to the DMA bit in the SPI DMA Control register. In this mode, the SPI controller transmits or receives the number of packets as indicated by the BLOCK_SIZE field in the SPI Block Size register.

If the PACKED bit is set and BIT_LEN is set to 7, then all FIFO words contain 4 packets to transfer (transmit or receive). Packets will be transferred as per the En_LE_Bit and En_LE_Byte bit configurations (see the En_LE_Bit and En_LE_Byte Modes section), with packet 0 in byte 0 of the FIFO and packet 3 in byte 3 of the FIFO.

In Unpacked Mode, if BIT_LEN is set to N, each packet will consist of (N + 1) bits. These bits will be transmitted/received in the Tx_FIFO/Rx_FIFO as per the En_LE_Bit and En_LE_Byte bit configurations (see the En_LE_Bit and En_LE_Byte Modes section). Any remaining bits in the FIFO will be ignored by the hardware. The maximum packet length is 32, which can be selected by setting BIT_LEN to 31. In this case, all data bits in the FIFO contain valid packet data.

A DMA request will be generated to APB_DMA in this mode depending on the setting of Tx_TRIG and Rx_TRIG. If transmits are enabled, setting Tx_TRIG to 00 will generate a Tx DMA request whenever the Tx FIFO has one word of space available (if not full). Setting Tx_TRIG to 01 will generate a Tx DMA request whenever the Tx FIFO has 4 words of space available. If receives are enabled, setting Rx_TRIG to 00 will generate an Rx DMA request whenever the Rx FIFO has one word of data available (is not empty). Setting Rx_TRIG to 01 will generate an Rx DMA request whenever the Rx FIFO has 4 words of data available.

35.2.2 PIO Mode

This mode is enabled by writing 1 to the PIO bit in the SPI Command1 register. In this mode, the SPI controller transmits/ receives as many packets as configured by software in the SPI Block Size Register.

PIO Mode has the same features as DMA Mode. The difference between PIO Mode and DMA Mode is that in PIO Mode, the maximum number of packets that can be transmitted or received is less than or equal to 64.

35.2.3 Interrupt Generation

The SPI controller generates an interrupt to the processor at the end of a transfer in PIO Mode or when an error is detected (both in PIO and DMA Modes) if the IE.TX or IE.RX bit in the SPI DMA Control register is enabled for transmit and receive modes of operation, respectively.

If IE.TX is enabled during transmits, an interrupt is generated whenever RDY or one of the FIFO status bits in the SPI FIFO Control/Status Register is set by the hardware. During transmits, when the SPI controller is configured as a Slave, if software/APB DMA cannot fill the transmit FIFO fast enough, hardware will set the Tx_FIFO_UNR bit and an underrun condition is generated. If software tries to write to a full Tx FIFO, hardware sets the Tx_FIFO_OVF bit as an indication that software attempted to overflow the Tx FIFO. Hardware makes sure that the overflowing data is never written to the Tx FIFO.

If IE.RX is enabled during receives, an interrupt is generated whenever RDY or one of the status bits in the SPI FIFO Control/Status Register is set by the hardware. During receives, when the SPI controller is configured as a Slave, if software/APB DMA cannot read the receive FIFO fast enough, hardware will set the Rx_FIFO_OVF bit and an overflow condition is generated. However, if software tries to read from an empty Rx FIFO, hardware sets the RXF_UNR bit as an indication that software attempted to underrun the Rx FIFO.

The interrupt can be cleared by writing a 1 to the source of the interrupt. If the interrupt is generated by assertion of RDY, then writing a 1 to the RDY bit clears the interrupt. If the interrupt is generated by assertion of Tx_FIFO_OVF, then writing a 1 to the TXF_OVF bit clears the interrupt. If the interrupt is generated by assertion of Rx_FIFO_UNR, then writing a 1 to RXF_UNR bit clears the interrupt.

> **Guideline for DMA Mode use cases**: In DMA Mode, if interrupts are enabled, the SPI software driver has to deal with 3 interrupts: one from the SPI controller and two from the Tx and Rx APB DMA channels. This might make it complicated for the driver and also stresses the system. In such cases, software can use the Rx DMA interrupt as the final one at which point all hardware activities can be assumed to be complete. But, in variable transfer lengths (continuous mode), software has to rely on the controller's interrupts.

35.2.4 Clock Initialization and Control

The SPI controller runs at 50 MHz at its interface to external SPI devices. The internal SPI controller clock (spi_clk) should run at the same frequency as the outgoing Interface Clock (Sclk_Out) in Master Mode. In Slave Mode, the internal SPI interface clock frequency has to be 50% greater than the external clock (Sclk_in).

The SPI clock is selected from different clock sources using a generic clock switch module. The clock switch module supports glitch-free switching of clock from one source to another. There are 4 clock sources:

- osc_clk
- pllP_out
- pllC_out
- pllM_out0

The selected clock is divided by an 8-bit value written in the divider register. The clock is then trimmed, and gated branches are generated for the SPI instances.

Note: The ratio between the slave core clock and the slave interface clock must be maintained between 1.5x and 4x. For example if the slave interface clock is running at 10 MHz, the slave core clock must not be less than 10x1.5 MHz and not greater than 10x4 MHz.

The CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0, CLK_RST_CONTROLLER_CLK_OUT_ENB_V_0, and CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0 registers in the CAR (Clock and Reset) block contain the enable bits for the SPI controller. In addition, the clock source and divider registers (CLK_SOURCE_SBC1, CLK_SOURCE_SBC2, CLK_SOURCE_SBC3, CLK_SOURCE_SBC4, CLK_SOURCE_SBC5, and CLK_SOURCE_SBC6) contain the 2-bit field SBCx_CLK_SRC to determine the PLL clock source, while the SBCx_CLK_DIVISOR field determines the divide ratio.

See the Clock and Reset Controller section for more information on clock configuration.

Controller Reset

The SPI controller can be reset by writing 1 to SPI1 (bit 11) in the RST_DEVICES.H (0x6000:6008) register. Software needs to write a 0 to this bit to bring the SPI controller out of reset.

See the Clock and Reset Controller section for more information on SPI controller reset.

Slave Mode GPIO Synchronization

Because SPI does not support flow control, the GPIO is required to have the proper communication to avoid any clock loss from the Master. GPIO can be used in the following way for proper synchronization:

- SPI slave client software will write to the Config registers of the SPI controller and make the SPI slave ready.
- After that the slave client software will toggle GPIO to inform the Master that the Slave is ready.
- Then the SPI master sends SCLK to the slave device.

If there is no GPIO to tell the Master that the Slave is ready, there is a chance to lose clock/data.

Guidelines

This section provides guidelines for programming the SPI controller:

- Program all the required register fields (no particular order is required except for the PIO/DMA bit, which has to be programmed last):
  - Clock Mode
  - Packed/Unpacked Mode
  - Tx_EN/Rx_EN

- BOTH_EN_BYTE/BOTH_EN_BIT
- En_LE_Bit/En_LE_Byte
- BIT_LEN and BLOCK_SIZE values
- CS_SW_HW (software based or hardware based)
  - If CS_SW_HW is set, the value on CS_SW_VAL will be driven out.
  - When CS HW is used program the Setup & Hold Values in CS Timing Register.
- Program the DMA Trig values appropriately, if DMA Mode is used
- Enable interrupts if PIO Mode is used

- Lastly, program PIO or DMA to indicate the start of transfer.

Once software enables PIO/DMA, no required bits can be changed until the end of transfer except for PIO/DMA bit. Clearing the PIO/DMA bit will end the transaction. In case the SPI controller is configured to Receive Mode and software clears the PIO/DMA bit, then the partial data which the controller received until then will be written into the FIFO. This is true both for Master and Slave.

Do not enable conflicting features. For example, avoid enabling BOTH_EN_BIT (or) BOTH_EN_BYTE along with bidirectional (BIDIR) mode.

In Slave Mode after PIO or DMA is set, hardware first waits for the CS to go low by one of the 4 Masters. Then on reception of Sclk, transfer begins. If CS or Sclk is removed by the Master in the middle of a transfer, software must terminate the transaction by clearing the PIO/DMA bit. In case of Rx, hardware then writes the last received packet (incomplete packet) to the Rx FIFO. If software does not clear the PIO/DMA bit, the SPI controller will just keep waiting for the Sclk from the Master. If Sclk resumes later, the transaction continues from where it paused earlier.

If any error conditions occur, hardware will set the corresponding status bits in the SPI FIFO Control/Status register and stop the transfer. If IE is enabled, an interrupt is generated. Software has to clear the source of the interrupt by writing a 1 to it, after the completion of ISR.

Special Guidelines for Slave Mode

Software should not program the controller register when the interface clock is toggling. Software can disable the external clock input before programming the registers and then re-enable it once all the registers bits are programmed.

Error Conditions

When the SPI controller is configured as a Master, the following error scenarios can happen:

- Tx_FIFO overflow

  Tx_FIFO overflow happens when the CPU/APB_DMA writes into the Tx FIFO when it is full. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with the Tx_FIFO_OVF bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and the Tx_FIFO_OVF bit. To start a new transaction, software has to flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.
- Rx_FIFO overflow

  Rx_FIFO overflow happens when the CPU/APB_DMA writes into the Rx FIFO when it is full in BOTH_EN Mode. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with Rx_FIFO_OVF bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and the Rx_FIFO_OVF bit. To start a new transaction, software has to flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.
- Tx_FIFO underrun

Tx_FIFO underrun happens when the CPU/APB_DMA reads from the Tx FIFO when it is empty. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with the Tx_FIFO_UNR bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and the Tx_FIFO_UNR bit. To start a new transaction, software has to flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

- Rx_FIFO underrun

  Rx_FIFO underrun happens when the CPU/APB_DMA reads from the Rx FIFO in BOTH_EN Mode. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with the Rx_FIFO_UNR bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and the Rx_FIFO_UNR bit, along with the Interrupt bit. To start a new transaction, software has to flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

  Note: In Master Mode, all the errors above can happen only when the CPU/APB_DMA reads/writes an empty/full FIFO. The SPI controller will never write/read a full/empty FIFO. Instead the controller will pause (stops sending clocks to the slave with chip select being in an active state) whenever the FIFOs are full/empty.

When the SPI controller is configured as a Slave, the following error scenarios can happen:

- Tx_FIFO overflow

  Tx_FIFO overflow happens when the CPU/APB_DMA or the SPI controller writes into the Tx FIFO when it is full. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with Tx_FIFO_OVF bit in the status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and Tx_FIFO_OVF bit. To start a new transaction, software has to disable the Master from sending clocks and then flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

- Rx_FIFO overflow

  Rx_FIFO overflow happens when the CPU/APB_DMA or the SPI controller writes into Rx FIFO when it is full in BOTH_EN Mode. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with Rx_FIFO_OVF bit in the status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and Rx_FIFO_OVF bit along with the Interrupt bit. To start a new transaction, software has to disable the Master from sending clocks and flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

- Tx_FIFO underrun

  Tx_FIFO underrun happens when the CPU/APB_DMA or the SPI controller reads from Tx FIFO when it is empty. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with the TX_FIFO_UNR bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and TX_FIFO_UNR bit along with the Interrupt bit. To start a new transaction, software has to disable the Master from sending clocks and flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

- Rx_FIFO underrun

  Rx_FIFO underrun happens when the CPU/APB_DMA or the SPI controller reads from the Rx FIFO in BOTH_EN Mode. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with the RX_FIFO_UNR bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and RX_FIFO_UNR bit along with the Interrupt bit. To start a new transaction, software has to disable the Master from sending clocks and flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

- CS_inactive

When the SPI controller is configured as a Slave and cs_active between packets is set, if an external Master deasserts the chip select in the middle of a transaction, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and sets the CS_INACTIVE bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the CS_INACTIVE bit. To start a new transaction, software has to disable the Master from sending clocks and flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

It is software's responsibility to read the data that is left out of the RX FIFO for CS inactive. For example, if the interrupt trigger is set for 4 words, and external master has taken the CS off in between the transfer, and, at that point of time, the RX FIFO has only 2 words, it is software's responsibility to read out those 2 words or flush the FIFO.

- Frame_End

  When the SPI controller is configured as a Slave and DMA-continuous and cs_active between packets is set, if an external Master stops sending clocks for more than slave_idle_clock_count programmed in the register, the SPI controller will end the transaction by setting the DMA bit to 0, and sets the FRAME_END bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the FRAME_END bit along with the Interrupt bit. To start a new transaction, software has to disable the Master from sending clocks and flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register. If the Master resumes the clk within "slave_idle_clock_count", the SPI controller will also pause and resume the transaction whenever ext_clk is available.

Programming Trimmers

The 3 programmable trimmers in the SPI controller are used only in Master Mode:

- SPI-Tx trimmer

  This trimmer is used in Master Tx mode to adjust/center the outgoing data with respect to the outgoing clock.

- SPI-Rx trimmer

  This trimmer is used in Master Rx mode to delay the loopback clock to the Rx shift registers. This trimmer is used to adjust the Master SCLK with respect to the SPI slave device's Tx data.

- SPI-Spare Control Register Byte1

  This 3-bit trimmer controls the internal timing of the Master mode Rx datapath. This trimmer along with the SPI Rx trimmer may be set independently to adjust the loopback clock delay. This trimmer adjusts the programmable delay on the u_spi_slave_rx* flops in inbound paths. These are half cycle paths, and delays have been added to increase Setup margin on these paths.

- The SPI controller has a programmable delay chain with a 3-tap delay chain. The 3 taps use 100D, 100D, and 60D and are programmed as follows:
  - 000: Bypass through each mux: will see 3 muxes in path
  - 001: 60D
  - 010: 100D
  - 100: 100D
  - 011: 60D + 100D
  - 110: 100D + 100D
  - 101: 100D+60D
  - 111: 100D+100D+60D
- Spare cells used as control for these delay taps:
  - (u_spi1_shift_padmacro/spi_spare_control_byte1_in_reg_2_,
  - u_spi1_shift_padmacro/spi_spare_control_byte1_in_reg_1_,

- u_spi1_shift_padmacro/spi_spare_control_byte1_in_reg_0_)

35.3 SPI Controller Registers

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

There are six sets of the SPI controller registers, one for each of the six SPI controllers. The register descriptions below give the offset of each register within its SPI controller’s address range. See the Address Map section for the starting address of each SPI controller.

35.3.1 SPI_COMMAND1

SPI Command1 Register

This register is used to set the bit length and to select the transfer mode.

Chip Select can also be selected to be in hardware mode as software mode with both active high and active low polarities to support devices with varying CS polarities.

Offset: 0x0 │ Read/Write: R/W │ Reset: 0x43d00000 (0b01000011110100000000xxxxx000000)

Bit	Reset	Description
31	0x0	PIO: Program 1 after all the other bits in the SPI_COMMAND2 and SPI_COMMAND1 registers are programmed to start the transfer. Hardware clears this bit automatically after the transfer is done. Clearing of this bit by software will stop the shifter and latch the partial data into the buffer (in Receive Mode). 0 = STOP (default) 1 = GO
30	0x1	M/S: Master/Slave mode select. 0 = Slave Mode (external clock) 1 = Master Mode (internal clock) (default)
29:28	0x0	MODE: The SPI interface clock mode has to be programmed according to the device with which it is communicating. 0 = Mode 0 (default) 1 = Mode 1 2 = Mode 2 3 = Mode 3 Master: All modes are supported both for Tx and Rx. Slave: Modes 1and 3 are supported for Tx Slave: All modes are supported for Rx.
27:26	0x0	CS_SEL: In Master Mode, these bits are programmed to select a slave in the multi-slave environment. 0 = Selects CS0 (default) 1 = Selects CS1 2 = Selects CS2 3 = Selects CS3
25	0x1	CS_POL_INACTIVE#3: In Master or Slave Mode, the inactive value of the external device’s cs value, which is connected to CS3, needs to be programmed. 1 = CS3 Inactive value of External device is high, 0 = CS3 Inactive value of External device is low 0 = LOW 1 = HIGH (default)
24	0x1	CS_POL_INACTIVE#2: In Master or Slave Mode, the inactive value of the external device’s cs value, which is connected to CS2, needs to be programmed, 1 = CS2 Inactive value of External device is high, 0 = CS2 Inactive value of External device is low 0 = LOW 1 = HIGH (default)
23	0x1	CS_POL_INACTIVE#1: In Master or Slave Mode, the inactive value of the external device’s cs value, which is connected to CS1, needs to be programmed. 1 = CS1 Inactive value of

Bit	Reset	Description
		External device is high, 0 = CS1 Inactive value of External device is low 0 = LOW 1 = HIGH (default)
22	0x1	CS_POL_INACTIVE#0: In Master or Slave Mode, the inactive value of the external device's cs value, which is connected to CS0, needs to be programmed. 1 = CS0 Inactive value of External device is high, 0 = CS0 Inactive value of External device is low 0 = LOW 1 = HIGH (default)
21	0x0	CS_SW_HW: Software control of the SPI_CS signal in Master Mode. In Slave Mode, this bit need not be programmed. 1 = CS controlled by software; 0 = CS controlled by hardware 0 = SPI _CS#(CS_SEL) is driven to the active state during packet transfers by the hardware (default) 1 = SPI _CS#(CS_SEL) is driven with the value in the CS_SW_VAL bit
20	0x1	CS_SW_VAL: CS signal value in Master Mode. In Slave Mode, this bit need not be programmed. If CS_SW_HW is 1, then the value in CS_SW_VAL is driven out on CS. If CS_SW_HW is 0, then this bit has no effect. 0 = CS is LOW 1 = CS is HIGH (default)
19:18	0x0	IDLE.SDA: Inactive data signal format. Controls the output enable of the SDA line when the controller is inactive and not doing data transfers. 0 = Drive low. (Master Mode) (default) 1 = Drive high. (Master Mode) 2 = External Pull Down. (Slave Mode) 3 = External Pull high. (Slave Mode)
17	0x0	BIDIR: Bidirectional Transfer Control Bit: This bit enables the bidirectional mode of the transfer. 0 = Normal Mode (default) 1 = Bidirectional Mode
16	0x0	En_LE_Bit: 1 = Enable Little Endian Bit. 0 = Disable Little Endian Bit 0 = LAST 1 = FIRST
15	0x0	En_LE_Byte: 1 = Enable Little Endian Byte. 0 = Disable Little Endian Byte 0 = LAST 1 = FIRST
14	0x0	BOTH_EN_BIT (x2 Mode): Both MISO and MOSI can also be used to transmit or receive at the same time when this bit is set. Even bits from the Tx FIFO packet are transmitted/received on MOSI, and Odd bits from the Tx FIFO packet are transmitted/received on MISO. Refer to the Both Enable Byte and Bit Modes (SPI x2 Mode) section for more information. 0 = Normal Mode (default) 1 = Transmit/Receive on both MISO and MOSI at the same time when the Tx_EN and Rx_EN bits are set
13	0x0	BOTH_EN_BYTE (x2 Mode): The 2 FIFOs can also be used to transmit or receive at the same time when this bit is set. Refer to the Both Enable Byte and Bit Modes (SPI x2 Mode) section for more information. **Note**: the BIDIR bit has to be 0 before this bit is set. 0 = Normal Mode (default) 1 = Transmit/Receive on both MISO and MOSI at the same time when the Tx_EN and Rx_EN bits are set
12	0x0	Rx_EN: Receive enable. Setting this bit to 1 enables the controller to receive the data. 0 = DISABLE (default) 1 = ENABLE
11	0x0	Tx_EN: Transmit enable. Setting this bit to 1 enables the controller to transmit the data. 0 = DISABLE (default) 1 = ENABLE
10:6	N/A	Reserved for future use. Always write zeros.

Bit	Reset	Description
5	0x0	PACKED: Packed mode enable bit.. 0 = Unpacked Mode (default) 1 = Packed Mode. This is only valid if the BIT_LEN field is set to either 3 (4-bit transfer), 7 (8-bit transfer), 15 (16-bit transfer), or 31 (32-bit transfer).
4:0	0x0	BIT_LEN: This field is used during PIO and DMA transfers. It represents the number of bits transmitted/received in either Packed or Unpacked Mode. The minimum bit length in Master Mode is 4 and the minimum length in Slave Mode is 1 byte (BIT_LEN = 7). The default is 0. 3 = 4-bit transfer N = N+1-bit transfer 31 = 32-bit transfer

35.3.2 SPI_COMMAND2

SPI Command2 Register

Offset: 0x4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
31:12	N/A	Reserved for future use. Always write zeros.
11:6	0x0	Tx_Clk_TAP_DELAY: Delays the clock going out to the external device with these tap values. Useful only in Master Mode.
5:0	0x0	Rx_Clk_TAP_DELAY: Delays the clock coming in from the external device with these tap values. Useful only in Master Mode.

35.3.3 SPI_CS_TIM1

SPI CS Timing1 Register

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

Bit	Reset	Description
31:28	0x0	CS_SETUP_TIME#3: Specifies the setup time of the chip select to the data being transmitted or received on CS#3 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
27:24	0x0	CS_HOLD_ TIME#3: Specifies the hold time of the chip select to the data being transmitted or received on CS#3 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
23:20	0x0	CS_SETUP_ TIME#2: Specifies the setup time of the chip select to the data being transmitted or received on CS#2 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
19:16	0x0	CS_HOLD_ TIME#2: Specifies the hold time of the chip select to the data being transmitted or received on CS#2 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
15:12	0x0	CS_SETUP_ TIME#1: Specifies the setup time of the chip select to the data being transmitted or received on CS#1 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.

Bit	Reset	Description
11:8	0x0	CS_HOLD_ TIME#1: Specifies the hold time of the chip select to the data being transmitted or received on CS#1 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
7:4	0x0	CS_SETUP_ TIME#0: Specifies the setup time of the chip select to the data being transmitted or received on CS#0 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
3:0	0x0	CS_HOLD_ TIME#0: Specifies the hold time of the chip select to the data being transmitted or received on CS#0 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.

35.3.4 SPI_CS_TIM2

SPI CS Timing2 Register

Offset: 0xc │ Read/Write: R/W │ Reset: 0x20202020 (0bxx100000xx100000xx100000xx100000)

Bit	Reset	Description
31:30	N/A	Reserved for future use. Always write zeros.
29	0x1	CS_ACTIVE_BETWEEN_PACKETS#3: Specifies if CS stays active between two packets on CS#3 1 = CS active between two packets (default) 0 = CS inactive between two packets
28:24	0x0	CYCLES_BETWEEN_PACKETS#3: Specifies the number of cycles between two packets in the PIO/DMA Mode for communication on CS#3. The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
23:22	N/A	Reserved for future use. Always write zeros.
21	0x1	CS_ACTIVE_BETWEEN_PACKETS#2: Specifies if CS stays active between two packets on CS#2 1 = CS active between two packets (default) 0 = CS inactive between two packets
20:16	0x0	CYCLES_BETWEEN_PACKETS#2: Specifies the number of cycles between two packets in the PIO/DMA Mode for communication on CS#2. The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
15:14	N/A	Reserved for future use. Always write zeros.
13	0x1	CS_ACTIVE_BETWEEN_PACKETS#1: Specifies if CS stays active between two packets on CS#1 1 = CS active between two packets (default) 0 = CS inactive between two packets
12:8	0x0	CYCLES_BETWEEN_PACKETS#1: Specifies the number of cycles between two packets in the PIO/DMA Mode for communication on CS#1. The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
7:6	N/A	Reserved for future use. Always write zeros.
5	0x1	CS_ACTIVE_BETWEEN_PACKETS#0: Specifies if CS stays active between two packets on CS#0

Bit	Reset	Description
		1 = CS active between two packets (default) 0 = CS inactive between two packets
4:0	0x0	CYCLES_BETWEEN_PACKETS#0: Specifies the number of cycles between two packets in the PIO/DMA Mode for communication on CS#0. The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.

35.3.5 SPI_TRANS_STATUS

SPI TRANSFER Status Register

Read this register for the status of the transfer.

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x00ff0000 (0bx0xxxxx11111111xxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	N/A		Reserved for future use. Always write zeros.
30	RW	0x0	RDY: Ready bit. This bit is set to 1 at the end of every transfer, and an interrupt is also generated if the corresponding interrupt enable is set in PIO/DMA Mode. Software writes a 1 to clear it. The interrupt is also cleared when this bit is cleared. 0 = NOT_READY (default) 1 = READY
29:24	N/A		Reserved for future use. Always write zeros.
23:16	RW	0xff	SLV_IDLE_COUNT: Slave Continuous Mode. If Sclk is not received for this number of cycles, then Slave Continuous mode is terminated, and the status bit FRAME_END is set. The default is 255 (8'hff).
15:0	RO	X	BLOCK_COUNT: Counts the number of packets in a transaction (Tx or Rx) in DMA/PIO Mode. In DMA Continuous Mode, this field gives the number of packets only if the number of packet is less than 2^{16}.

35.3.6 SPI_FIFO_STATUS

SPI Control/Status FIFO Status Register

SPI STATUS register: Read this register to know the status of the transfer. Error bit is set whenever we encounter Errors such as Underflow/Overflow

Offset: 0x14 │ Read/Write: R/W │ Reset: 0x00400005 (0b00xxxxxxxxxxxxxx00xxxxx00000xxxx)

Bit	R/W	Reset	Description
31	RW	0x0	CS_INACTIVE: When the SPI is in Slave Mode, if CS is deasserted, this bit is set to indicate an error and the received number of bits is written in the Rx FIFO. An interrupt is generated if IE.Tx or IE.Rx is set. The default is 0.
30	RW	0x0	FRAME_END: When the SPI is in Slave Continuous mode, if Sclk is not received for the number of clocks specified in the SLV_IDLE_COUNT field, the continuous mode is terminated and this status bit is set. The default is 0.
29:23	RO	X	RX_FIFO_FULL_COUNT: Indicates the number of slots in the receive FIFO remaining before the FIFO is empty. This field is used by software for debugging purposes.
22:16	RO	X	TX_FIFO_EMPTY_COUNT: Indicates the number of slots in the transmit FIFO remaining before the FIFO is full. This field is used by software for debugging purposes.
15	RW	0x0	RX_FIFO_FLUSH: Software writes a 1 to this bit to flush the Rx FIFO. This bit reads as 1 when the flush operation is in progress and returns to 0 when it is finished. 0 = NOP (default) 1 = FLUSH

Bit	R/W	Reset	Description
14	RW	0x0	TX_FIFO_FLUSH: Software writes a 1 to this bit to flush the Tx FIFO. This bit reads as 1 when the flush operation is in progress and returns to 0 when it is finished. 0 = NOP (default) 1 = FLUSH
13:9	N/A		Reserved for future use. Always write zeros.
8	RW	0x0	ERR: Will be set to 1 by hardware when errors such as underflow/overflow occur. Software writes a 1 to clear the flag. 0 = OK (default) 1 = ERROR
7	RW	0x0	TX_FIFO_OVF: TX FIFO Overflow. This bit is set to 1 whenever the SPI controller or software tries to write to a full Tx FIFO. An interrupt is generated if the interrupt enable is set for transmit operations (IE.TX in the SPI_DMA_CTL register). Software writes a 1 to clear this bit. 0 = OK (default) 1 = ERROR
6	RW	0x0	TX_FIFO_UNR: TX FIFO Underrun. This bit is set to 1 whenever the SPI controller or software tries to read from an empty Tx FIFO. An interrupt is generated if the interrupt enable is set for transmit operations (IE.TXC in the SPI_DMA_CTL register). Software writes a 1 to clear this bit. 0 = OK (default) 1 = ERROR
5	RW	0x0	RX_FIFO_OVF: RX FIFO Overflow. This bit is set to 1 whenever the SPI controller or software tries to write into a full Rx FIFO. An interrupt is generated if the interrupt enable is set for receive operations (IE.RXC in the SPI_DMA_CTL register). Software writes a 1 to clear this bit. 0 = OK (default) 1 = ERROR
4	RW	0x0	RX_FIFO_UNR: RX FIFO Underrun. This bit is set to1 whenever the SPI controller or software tries to read from an empty Rx FIFO. An interrupt is generated if the interrupt enable is set for receive operations (IE.RXC in the SPI_DMA_CTL register). Software writes a 1 to clear this bit. 0 = OK (default) 1 = ERROR
3	RO	X	TX_FIFO_FULL: TX FIFO Full Status. Hardware sets this bit to 1 if the Tx FIFO is full; otherwise this bit is 0. The Tx FIFO is empty at power on reset (POR). 0 = NOT_FULL (default) 1 = FULL
2	RO	X	TX_FIFO_EMPTY: TX FIFO Empty Status. Hardware sets this bit to 1 if the Tx FIFO is empty; otherwise this bit is 0. The Tx FIFO is empty at POR. 0 = NOT_EMPTY 1 = EMPTY (default)
1	RO	X	RX_FIFO_FULL: RX FIFO Full Status. Hardware sets this bit to 1 if the Rx FIFO is full; otherwise this bit is 0. The Rx FIFO is empty at POR. 0 = NOT_FULL (default) 1 = FULL
0	RO	X	RX_FIFO_EMPTY: RX FIFO Empty Status. Hardware sets this bit to 1 if the Rx FIFO is empty; otherwise this bit is 0. The Rx FIFO is empty at POR. 0 = NOT_EMPTY 1 = EMPTY (default)

35.3.7 SPI_TX_DATA

SPI Transmit Data Register

Offset: 0x18 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description

31:0	X	SPI_Tx_DATA. Transmit Data. This register holds the last data that was transmitted by the SPI controller.

35.3.8 SPI_RX_DATA

SPI Receive Data Register

Offset: 0x1c │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SPI_Rx_DATA. Receive Data. This register holds the last data that was received by the SPI controller.

35.3.9 SPI_DMA_CTL

SPI DMA Control Register

SPI DMA Control Register: Used in the DMA transfers. We can set trigger levels in this register

Offset: 0x20 │ Read/Write: R/W │ Reset: 0x00000000 (0b0000xxxxxxx00xx00xxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	DMA: Enable DMA Mode transfer. Software writes a 1 to this bit to start a transfer in the DMA mode. All fields in the SPI_COMMAND1 and SPI_DMA_CTL register must be set before writing a 1 to this bit. This bit is cleared by the SPI controller after all packets have been transferred as indicated by the DMA_BLOCK_SIZE field. Clearing this bit by software will stop the shifter and latch the partial data into buffer. 0 = DMA Mode is disabled (default) 1 = DMA Mode is enabled
30	0x0	CONT: Enable Continuous Mode transfer. 0 = Continuous Mode is disabled (default) 1 = Continuous Mode is enabled
29	0x0	IE.RX: Interrupt enable on receive completion. 0 = Disable interrupt generation for receives (default) 1 = Enable interrupt generation at the end of a receive transfer
28	0x0	IE.TX: Interrupt enable on transmit completion. 0 = Disable interrupt generation for transmits (default) 1 = Enable interrupt generation at the end of a transmit transfer
27:21	N/A	Reserved for future use. Always write zeros.
20:19	0x0	RX_TRIG: Receive FIFO trigger level. 00: 1 word. DMA trigger is asserted whenever there is at least 1 packet in the RX FIFO. (default) 01: 4 words. DMA trigger is asserted when there are at least 4 packets in the RX FIFO. 10: 8 words. DMA trigger is asserted when there are at least 8 packets in the RX FIFO. 11: 16 words. DMA trigger is asserted when there are at least 16 packets in the RX FIFO. (Presently, the APB DMA does not support this trigger level.)
18:17	N/A	Reserved for future use. Always write zeros.
16:15	0x0	TX_TRIG: Transmit FIFO trigger level. 00: 1 word. DMA trigger is asserted whenever there is space for at least 1 packet in the TX FIFO. (default) 01: 4 words. DMA trigger is asserted when there is space for at least 4 packets in the TX FIFO. 10: 8 words. DMA trigger is asserted when there is space for at least 8 packets in the TX FIFO. 11: 16 words. DMA trigger is asserted when there is space for at least 16 packets in the TX FIFO. (Presently, the APB DMA does not support this trigger level.)

35.3.10 SPI_DMA_BLK

SPI Block Size Register

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31:16	N/A	Reserved for future use.
15:0	0x0	BLOCK_SIZE: Size of data block to be transferred in PIO/DMA Mode. The default is 0. In DMA Mode, the maximum number of 32-bit packets that can be transferred is 2^{16} = 65536. In PIO Mode, the maximum number of 32-bit packets that can be transferred is 64 (FIFO depth). The Block Size has to be programmed 1 packet lower than intended; for example, a value of 3 indicates 4 packets are to be transferred.

35.3.11 SPI_FIFO1

SPI TX FIFO Register

Offset: 0x108 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SPI Tx_FIFO. Tx FIFO.

35.3.12 SPI_FIFO2

SPI RX FIFO Register

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

Bit	Reset	Description
31:0	0x0	SPI Rx_FIFO. Rx FIFO.

35.3.13 SPI_SPARE_CTLR

SPI Spare Control Register

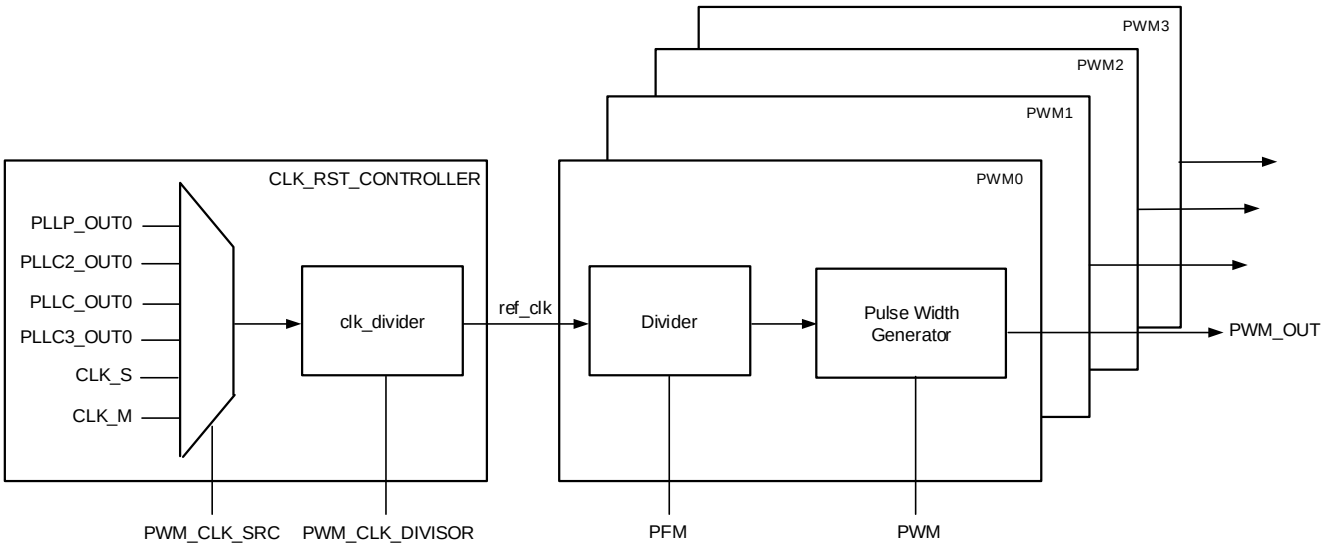
Offset: 0x18c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:11	0x0	Reserved for future use, always write zero.
10:8	0x0	SPI_SPARE_CONTROL_REGISTER_BYTE2. Program these bits with Rx_Clk_TAP_DELAY in the COMMAND2 register to adjust the clock delay on internal registers. Useful in Master Mode only.
7:0	0x0	Reserved for future use, always write zero.

36.0 PWM CONTROLLER

The Pulse Width Modulator (PWM) controller is a four channel frequency divider whose pulse width varies. Each channel has a programmable frequency divider and a programmable pulse width generator.

Figure 134: PWM Controller Block Diagram



Fpwm_out= Fref_clk / (PFM+1) / 256

Frequency division is a 13-bit programmable value, and pulse division is an 8-bit value.

The PWM controller runs off a device clock, which is programmed in the Clock and Reset controller, and can be any frequency up to the device clock maximum speed of 48 MHz.

The device clock frequency is always subject to a minimum divide by 256 to generate the PWM output frequency, and may also be subdivided to slow it down further based on the programmable value locally.

There is an APB interface that interfaces the register logic to the APB bus.

The PWM controller 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.

36.1 Functionality

There are four PWM controllers on four individual pins on the chip. The pinmux corresponding to these are SDC primary pin groups. For the four PWM controllers there are four corresponding registers:

- PWM_CSR0
- PWM_CSR1
- PWM_CSR2
- PWM_CSR3

Each PWM controller must be enabled (bit [31]) to be operational. Pulse Width [23:16] determines the output pulse width and must be programmed appropriately. Pulse Width [30:24] is not used and must be 0. Frequency Divider [12:0] determines the Divided clock.

36.2 PWM Registers

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

36.2.1 PWM_CONTROLLER_PWM_CSR_0_0

PWM Output-0 Configuration Control Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000xxx0000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable Pulse width modulator 0 = DISABLE 1 = ENABLE
30:16	0x0	PWM_0: pulse width that needs to be programmed. 0=Always low. 1=1/256 Pulse high. 2=2/256 Pulse high. N=N/256 Pulse high
12:0	0x0	PFM_0: Frequency divider that needs to be programmed.

36.2.2 PWM_CONTROLLER_PWM_CSR_1_0

PWM Output-1 Configuration Control Register

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000xxx0000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable pulse width modulator 0 = DISABLE 1 = ENABLE
30:16	0x0	PWM_0: pulse width that needs to be programmed. 0=Always low. 1=1/256 Pulse high. 2=2/256 Pulse high. N=N/256 Pulse high
12:0	0x0	PFM_1: Frequency divider that needs to be programmed.

36.2.3 PWM_CONTROLLER_PWM_CSR_2_0

PWM Output-2 Configuration Control Register

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000xxx0000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable pulse width modulator 0 = DISABLE 1 = ENABLE
30:16	0x0	PWM_0: Pulse width that needs to be programmed. 0=Always low. 1=1/256 Pulse high 2=2/256 Pulse high N=N/256 Pulse high
12:0	0x0	PFM_2: Frequency divider that needs to be programmed.

36.2.4 PWM_CONTROLLER_PWM_CSR_3_0

PWM Output-3 Configuration Control Register

Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000xxx0000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable pulse width modulator 0 = DISABLE 1 = ENABLE
30:16	0x0	PWM_0: Pulse width that needs to be programmed 0=Always low 1=1/256 Pulse high 2=2/256 Pulse high N=N/256 Pulse high
12:0	0x0	PFM_3: Frequency divider that needs to be programmed.

[THIS PAGE INTENTIONALLY LEFT BLANK]

37.0 THERMAL SENSOR AND THERMAL THROTTLING CONTROLLER

37.1 Thermal Throttling Controller (SOC_THERM)

The goals of the thermal throttling controller include:

- Thermal sensing: Access, capture and processing data from thermal sensors which are located in multiple locations around the SOC.
- Thermal throttling: Providing a means of reducing the chip performance to manage power consumption in response to thermal events. Provide support for several possible software and hardware throttling mechanisms.
- External OC Alarm Slowdown: Providing a means of reducing chip performance to manage power consumption on detecting over-current alarms from outside the Tegra® K1 SOC. Examples of such OC alarms include “AC power unplug” event, PMIC over-current, battery over-current, radio power amplifier over-current events. Some events require the system to be throttled to lower power/performance within latencies that cannot be handled by software alone.
- Throttling in a di/dt safe manner: Providing hardware support to ensure that throttling in response to over-temperature and over-current events happens in a di/dt safe manner. The support is implemented in the SOC_THERM (for pulse skipper) and in CAR (for PLL slowdown).

The intention of this section is not to provide a full programmer’s guide, but solely to document the register interface to allow better understanding of NVIDIA provided drivers.

37.2 Thermal Sensor

Thermal and voltage sensors are used to constantly monitor the temperature and voltage on the chip. Sensors are placed across the die to gauge the temperature of the whole chip. The TSensor module:

- Generates interrupt to software to lower temperature via DVFS, on reaching a certain thermal/voltage threshold
- Generates a signal to the CAR block to reduce CPU frequency by half, on reaching a certain thermal/voltage threshold
- Generates a signal to the PMC when temperature reaches dangerously high levels to reset the chip and sets a flag in PMC

37.3 Programming Guidelines

37.3.1 Boot-Time Initialization

At boot time, software needs to:

1. Program thermal thresholds. Thresholds for heavy hardware throttling and THERMTRIP should be in fuses.
2. Program CPU block activity weights.
3. Configure thermal and EDP throttling.
4. Initialize throttle settings.
5. Program lock registers to lock contents of important registers.
6. Initialize, set up, and train PLLs for clocking.
7. Enable thermal and EDP throttling.

37.3.2 LP1 Mode

Software should make sure that there is no active hardware throttling when entering LP1 mode.

On LP1 entry, the CPU_PWR_GOOD signal from the PMC to SOC_THERM is deasserted and the FCCPLEX is not throttled. Software needs to reprogram throttle vectors because the system generally runs at a lower frequency in this mode. It is not necessary to reprogram locked registers for transitioning into or out of LP1.

37.3.3 LP0 Mode

Software should disable throttling before entering LP0 mode to ensure that there is no ongoing hardware throttling and both PLL and pulse skippers are completely under software control.

Software needs to restore all control registers, thresholds, and throttle settings when exiting from LP0 mode. The exception to this is THERMTRIP_CTL which is handled by the boot ROM.

37.3.4 PLL Hardware Controller Sequences

Software needs to follow certain sequences when configuring the CPU PLL because the SOC_THERM can throttle the PLL.

37.3.4.1 PLLx Start-up Sequence

1. When PLLx is first enabled, software has full control. The default value of PLLx_ENABLE_SWCTL=1. The default hardware controller state is SEQ_OFF.
2. Software programs the relevant CAR registers to enable and train the PLL using the software bypass path.
3. Software programs default values to the PLLX_SW_RAMP_CFG registers.
4. Software programs PLLx_ENABLE_SWCTL=0 to relinquish software control of PLLx.

37.3.4.2 Software Use of the PLLx Hardware Controller

Software can use the PLLx Hardware Controller to ramp the PLLx in normal DVFS frequency change scenarios. Software has to use the following sequence to ramp the PLL using the hardware controller:

1. Check that the clock to the device is enabled via the CLK_OUT_ENB_L/H/U CAR registers.
2. Program software copies for DYNRAMP_STEPA, DYNRAMP_STEPB, NDIV_NEW, NDIV, and RAMP_MODE to the appropriate values.
3. Trigger the ramp by setting PLLx_SEQ_TRIGGER=1. If a ramp is already ongoing, the new values take effect after the current ramp is complete.
4. Software can monitor the PLLx_HW_CTRL_STATUS register to know the current ramp status and poll for SW_RAMP_DONE. In general, software uses this to determine if a corresponding voltage change is needed after the frequency change.
5. When SW_RAMP_DONE is asserted, software needs to clear PLLx_SEQ_TRIGGER.

37.3.4.3 Programming Sequence When Hardware Throttling is Engaged

Hardware throttling takes precedence, so the following sequence should hold even when software is trying to change frequency.

1. SOC_THERM_throttlectl asserts soc_therm2car_pllx_throttle_en=1.
   - The CAR asserts car2pllx_hw_throttle_en=1.
   - If PLLx_SW_OVERRIDE==0, then the internal hw_throttle_en signal is asserted.
2. SOC_THERM_throttlectl drives the dynamic ramp interface to the CAR for the intended target frequency.
3. SOC_THERM_throttlectl asserts either soc_therm2car_pllx_fast_mode or soc_therm2car_pllx_slow_mode.

4. SOC_THERM_throttlectl asserts soc_therm2car_pllx_seq_en=1.
   - The CAR asserts car2pllx_seq_en=1.
   - The programmed values are latched to their hardware copy, and a ramp sequence is started based on the selected mode.
   - If at this time, a ramp is already ongoing, then the new values take effect after the current ramp completes.
5. PLLx asserts PLL_LOCK=1 when the ramp is complete. This causes the sequencer to return to the SEQ_OFF state and assert pllx2car_ramp_done=1.
6. The CAR asserts car2soc_therm_pllx_ramp_done=1. This completes ramp and handshake with SOC_THERM_throttlectl. PLLx is now locked to the target frequency.
7. SOC_THERM_throttlectl can initiate a new ramp (using a different target frequency and mode) after this point.

37.3.4.4 Sequence when both Hardware and Software Try to Program the PLL

Hardware throttling is given precedence over software when ramping PLLx. This is done by providing a separate set of registers (labeled hardware copy in the block diagram) and separate set of ramp control signals. Hardware uses the hw_throttle_en and hw_seq_trigger signals while software uses the PLLx_SEQ_TRIGGER signal to start a new ramp. If hw_throttle_en==1, then the hardware throttle values are muxed into the PLL.

Note that there is no mechanism to stop and restart a sequence in the middle of a ramp. So if software has managed to start a ramp (in slow or fast mode), hardware throttling begins only after the current ramp is done. Hardware throttling has to tolerate this latency. The latency can be reduced if software uses a combination of fast and slow modes for this ramp.

37.3.4.5 Sequence when Software is Changing PLLs for a new PLLx VCO Range

If the target frequency chosen by SW DVFS is outside the current PLLx VCO range, then software first selects a different PLL, re-locks PLLx to the new range, and then switches back to PLLx.

For Outbound Path (Reprogramming PLLx to New Range):

Typically, software switches either to PLLP/2 (200 MHz) or PLLP (400 MHz) clock. To avoid di/dt issues with this switch, software should first do a dynamic ramp down to VCOmin (it can be fast ramp) and then switch to PLLP.

For Inbound Path (Switching Back to PLLx After Re-Locking to New Range):

Similarly, when the new VCO range is programmed, switching directly to the new target frequency may also be a di./dt event. To avoid di/dt problems, software should first switch to VCOmin of new range and dynamically ramp up to the target frequency.

The following sequence covers these cases:

1. Switch to the new PLL:
   - Program CAR PLLx registers to ramp down to VCOmin.
   - Set PLLx_ENABLE_SWCTL=1. This disables the PLLx hardware controller.
   - PLLx latches in the new settings using the software bypass path. PLLx switches to the new settings when the ongoing ramp (if any) is completed.
2. Software reads the status registers to check when the ramp is complete.
3. Use existing software ↔ CAR↔ PLLx path to disable PLLx.
4. Re-program PLLx.
   - Software programs PLL to VCOmin.
5. Switch back to PLLx.
   - Then dynamic ramp up to target – 80% fast, 20% slow.

6. Enable the hardware sequencer (see Enabling and Disabling the Hardware Sequencer). PLLx_ENABLE_SWCTL=0. Hardware cannot ramp PLLx until this step is done.

At this point, the PLLx can start ramping to the new frequency. If a hardware throttling event has occurred at the time when controller is enabled, the hardware values are used to ramp the PLLx.

37.3.4.6 Sequence when Software Wants to Switch PLLs

At low frequencies, software may choose to switch the clock source for a unit to a different PLL. In this case, software does not necessarily turn PLLx (or the hardware controller) OFF to ensure a fast response time when the clock source is switched back to PLLx. However, this presents a problem for hardware throttling because SOC_THERM does not have visibility of the clock source switch. In this case, SOC_THERM_throttlectl may continue to try and throttle PLLx. Since PLLx is enabled, this may cause PLLx to ramp to the throttled value. However, since it is not feeding into any unit, the ramp does not have any effect on the unit frequency. Other than wasting power, this does not have any problematic effect on the unit operation.

It is also useful if software can communicate to the SOC_THERM_throttlectl block when it switches PLLs. If SOC_THERM_throttlectl has this information and still requires throttling, then it will use only pulse skipping instead of trying to ramp the PLLx. The throttle sequencer configuration register DISABLE_PLLx_THROTTLE field can be used for this purpose.

When software wants to switch back to PLLx, it is likely that PLLx is not at the same frequency when software had switched clock sources. In this case, software has two options:

1. Use Software Override Mode**:** When software wants to switch back, it asserts the PLLx_SW_OVERRIDE bit so that hardware throttling is disabled. It can then choose either to program new values or use existing ones. In either case, software needs to set PLLx_SEQ_TRIGGER=1 to make sure these values take effect. The sequence is as follows:
   a. Set PLLx_SW_OVERRIDE=1.
   b. (optional) Program the PLLx_SW_CFG registers with target frequency values.
   c. Set PLLx_SEQ_TRIGGER=1.
   d. Wait for SW_RAMP_STATUS=1
      - If PLLx is currently in the middle of a hardware ramp, then SW_RAMP_STATUS is asserted after the software values have taken effect.
   e. Switch the clock source to PLLx.
2. Use Software Bypass Path: In this case, software disables the hardware controller and programs PLLx to a target frequency through the default CAR registers. It switches the clock source to PLLx when this ramp is complete. Note that software still has to go through the “disable hardware controller sequence”.

37.3.4.7 RAMP_DONE Response to Software

The RAMP_DONE signal indicates the completion of the dynamic ramp and serves as an acknowledgement that the corresponding ramp settings were successfully applied. It is possible that software updates the PLLx configuration registers, but before it can begin its ramp, a hardware throttle event occurs that results in PLLx being ramped to a different configuration. In this case, the RAMP_DONE ack from this (i.e., hardware) ramp should not serve as the ack for software because a different configuration was used.

A separate SW_RAMP_DONE signal solves this problem. PLLx_HW_CTRL keeps track of the source of the current PLL configuration and asserts the corresponding ramp_done signal when the ramp is complete.

37.3.4.8 Enabling and Disabling the Hardware Sequencer

Enabling the Hardware Controller

1. Software enables and trains PLLx for initial start-up. Software programs default values to the PLLx_SW_CFG register.

2. Software sets PLLx_ENABLE_SWCTL=0 to relinquish control to the hardware sequencer. Either software or hardware can now start a ramp sequence if required.

Disabling the Hardware Controller

There are two possibilities:

1. Software disables the Hardware controller because it wants to take control of PLLx:
    a. Read PLLx_HW_CTRL_STATUS to know the current status of the sequencer.
    b. Set PLLx_ENABLE_SWCTL=1. This enables the software bypass path and disables the hardware controller.
    c. If PLLx is in the middle of a ramp, the software bypass values take effect after the current ramp completes.

Note that software does not need to wait or read any status for this.

2. Software wants to turn the PLLx OFF:
    a. Read PLLx_HW_CTRL_STATUS to know the current status of the sequencer. Wait for RAMP_DONE=1 and hardware controller to go to SEQ_OFF state.
    b. Set PLLx_IDDQ_MODE=1 to turn OFF PLLx. This disables the dynamic ramp sequencer FSM.

When software wants to turn PLLx back on, it has to follow the sequence described in the PLLx Start-up Sequence subsection.

37.4 TSensor Registers

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

All SOC_THERM registers are in the APB address space. Refer to the Address Map section for the base addresses.

37.4.1 SOC_THERM_THERMCTL_LEVEL0_GROUP_CPU_0

Offset: 0x0 │ Read/Write: R/W │ Reset: 0x0000000X (0bxxxxxxx00000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:status

Bit	R/W	Reset	Description
			0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

37.4.2 SOC_THERM_THERMCTL_LEVEL0_GROUP_GPU_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

37.4.3 SOC_THERM_THERMCTL_LEVEL0_GROUP_MEM_0

Offset: 0x8 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE

Bit	R/W	Reset	Description
			1 = LITE 2 = HEAVY
1:0	RO	X	S: Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

37.4.4 SOC_THERM_THERMCTL_LEVEL0_GROUP_TSENSE_0

Offset: 0xc | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000x00000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S: Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

37.4.5 SOC_THERM_THERMCTL_LEVEL0_UP_STATS_0

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

Bit	Reset	Description
31:0	X	COUNT: Count for UP threshold breaches for this level used in the lab

37.4.6 SOC_THERM_THERMCTL_LEVEL0_DN_STATS_0

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

Bit	Reset	Description
31:0	X	COUNT: Count for DN threshold breaches for this level used in the lab

37.4.7 SOC_THERM_THERMCTL_LEVEL1_GROUP_CPU_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC

Bit	R/W	Reset	Description
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

37.4.8 SOC_THERM_THERMCTL_LEVEL1_GROUP_GPU_0

Offset: 0x24 │ Read/Write: R/W │ Reset: 0x0000000X (0bxxxxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

37.4.9 SOC_THERM_THERMCTL_LEVEL1_GROUP_MEM_0

Offset: 0x28 │ Read/Write: R/W │ Reset: 0x0000000X (0bxxxxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS:TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

37.4.10 SOC_THERM_THERMCTL_LEVEL1_GROUP_TSENSE_0

Offset: 0x2c │ Read/Write: R/W │ Reset: 0x0000000X (0bxxxxxxx00000000000000000x00000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

37.4.11 SOC_THERM_THERMCTL_LEVEL1_UP_STATS_0

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

Bit	Reset	Description
31:0	X	COUNT: count for UP threshold breaches for this level used in the lab

37.4.12 SOC_THERM_THERMCTL_LEVEL1_DN_STATS_0

Offset: 0x34 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: count for DN threshold breaches for this level used in the lab

37.4.13 SOC_THERM_THERMCTL_LEVEL2_GROUP_CPU_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

37.4.14 SOC_THERM_THERMCTL_LEVEL2_GROUP_GPU_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON

Bit	R/W	Reset	Description
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

37.4.15 SOC_THERM_THERMCTL_LEVEL2_GROUP_MEM_0

Offset: 0x48 │ Read/Write: R/W │ Reset: 0x0000000X (0bxxxxxxx00000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

37.4.16 SOC_THERM_THERMCTL_LEVEL2_GROUP_TSENSE_0

Offset: 0x4c │ Read/Write: R/W │ Reset: 0x0000000X (0bxxxxxxx00000000000000000x00000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON

Bit	R/W	Reset	Description
6:5	RW	0x0	CPU:Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU:Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

37.4.17 SOC_THERM_THERMCTL_LEVEL2_UP_STATS_0

Offset: 0x50 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Count for UP threshold breaches for this level used in the lab

37.4.18 SOC_THERM_THERMCTL_LEVEL2_DN_STATS_0

Offset: 0x54 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Count for DN threshold breaches for this level used in the lab

37.4.19 SOC_THERM_THERMCTL_LEVEL3_GROUP_CPU_0

Offset: 0x60 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:status 0 = BELLOW 1 = IN 2 = RES

Bit	R/W	Reset	Description
			3 = ABOVE

37.4.20 SOC_THERM_THERMCTL_LEVEL3_GROUP_GPU_0

Offset: 0x64 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

37.4.21 SOC_THERM_THERMCTL_LEVEL3_GROUP_MEM_0

Offset: 0x68 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS:TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU :Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY

Bit	R/W	Reset	Description
1:0	RO	X	S:status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

37.4.22 SOC_THERM_THERMCTL_LEVEL3_GROUP_TSENSE_0

Offset: 0x6c │ Read/Write: R/W │ Reset: 0x0000000X (0bxxxxxxx00000000000000000x00000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

37.4.23 SOC_THERM_THERMCTL_LEVEL3_UP_STATS_0

Offset: 0x70 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Count for UP threshold breaches for this level used in the lab

37.4.24 SOC_THERM_THERMCTL_LEVEL3_DN_STATS_0

Offset: 0x74 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Count for DN threshold breaches for this level used in the lab

37.4.25 SOC_THERM_THERMCTL_THERMTRIP_CTL_0

Offset: 0x80 │ Read/Write: R/W │ Reset: 0x10696969 (0bxxx10000011010010110100101101001)

Bit	Reset	Description
28	0x1	ANY_EN: Initiate THERMTRIP based on any monitoring group 0 = OFF 1 = ON

Bit	Reset	Description
27	0x0	MEM_EN: Initiate THERMTRIP based on MEM monitoring group 0 = OFF 1 = ON
26	0x0	GPU_EN: Initiate THERMTRIP based on GPU monitoring group 0 = OFF 1 = ON
25	0x0	CPU_EN: Initiate THERMTRIP based on CPU monitoring group 0 = OFF 1 = ON
24	0x0	TSENSE_EN: Initiate THERMTRIP based on TSENSE monitoring group
23:16	0x69	GPU_N_MEM: Threshold for thermal shutdown for GPU group
15:8	0x69	CPU: Threshold for thermal shutdown for CPU group
7:0	0x69	TSENSE: Threshold for thermal shutdown for TSENSE group

37.4.26 SOC_THERM_THERMCTL_INTR_STATUS_0

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

Bit	Reset	Description
31	0x0	MD3:MEM group down threshold interrupt for level 3 0 = OFF 1 = ON
30	0x0	MU3:MEM group up threshold interrupt for level 3 0 = OFF 1 = ON
29	0x0	MD2:MEM group down threshold interrupt for level 2 0 = OFF 1 = ON
28	0x0	MU2:MEM group up threshold interrupt for level 2 0 = OFF 1 = ON
27	0x0	MD1:MEM group down threshold interrupt for level 1 0 = OFF 1 = ON
26	0x0	MU1:MEM group up threshold interrupt for level 1 0 = OFF 1 = ON
25	0x0	MD0:MEM group down threshold interrupt for level 0 0 = OFF 1 = ON
24	0x0	MU0:MEM group up threshold interrupt for level 0 0 = OFF 1 = ON
23	0x0	GD3:GPU group down threshold interrupt for level 3 0 = OFF 1 = ON
22	0x0	GU3:GPU group up threshold interrupt for level 3 0 = OFF 1 = ON
21	0x0	GD2:GPU group down threshold interrupt for level 2 0 = OFF 1 = ON

Bit	Reset	Description
20	0x0	GU2:GPU group up threshold interrupt for level 2 0 = OFF 1 = ON
19	0x0	GD1:GPU group down threshold interrupt for level 1 0 = OFF 1 = ON
18	0x0	GU1:GPU group up threshold interrupt for level 1 0 = OFF 1 = ON
17	0x0	GD0:GPU group down threshold interrupt for level 0 0 = OFF 1 = ON
16	0x0	GU0:GPU group up threshold interrupt for level 0 0 = OFF 1 = ON
15	0x0	CD3:CPU group down threshold interrupt for level 3 0 = OFF 1 = ON
14	0x0	CU3:CPU group up threshold interrupt for level 3 0 = OFF 1 = ON
13	0x0	CD2:CPU group down threshold interrupt for level 2 0 = OFF 1 = ON
12	0x0	CU2:CPU group up threshold interrupt for level 2 0 = OFF 1 = ON
11	0x0	CD1:CPU group down threshold interrupt for level 1 0 = OFF 1 = ON
10	0x0	CU1:CPU group up threshold interrupt for level 1 0 = OFF 1 = ON
9	0x0	CD0:CPU group down threshold interrupt for level 0 0 = OFF 1 = ON
8	0x0	CU0:CPU group up threshold interrupt for level 0 0 = OFF 1 = ON
7	0x0	TD3:TSENSE group down threshold interrupt for level 3 0 = OFF 1 = ON
6	0x0	TU3:TSENSE group up threshold interrupt for level 3 0 = OFF 1 = ON
5	0x0	TD2:TSENSE group down threshold interrupt for level 2 0 = OFF 1 = ON
4	0x0	TU2:TSENSE group up threshold interrupt for level 2 0 = OFF 1 = ON
3	0x0	TD1:TSENSE group down threshold interrupt for level 1 0 = OFF

Bit	Reset	Description
		1 = ON
2	0x0	TU1:TSENSE group up threshold interrupt for level 1 0 = OFF 1 = ON
1	0x0	TD0:TSENSE group down threshold interrupt for level 0 0 = OFF 1 = ON
0	0x0	TU0:TSENSE group up threshold interrupt for level 0 0 = OFF 1 = ON

37.4.27 SOC_THERM_THERMCTL_INTR_EN_0

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

Bit	Reset	Description
31	0x0	MD3:MEM group down threshold interrupt enable for level 3 0 = OFF 1 = ON
30	0x0	MU3:MEM group up threshold interrupt enable for level 3 0 = OFF 1 = ON
29	0x0	MD2:MEM group down threshold interrupt enable for level 2 0 = OFF 1 = ON
28	0x0	MU2:MEM group up threshold interrupt enable for level 2 0 = OFF 1 = ON
27	0x0	MD1:MEM group down threshold interrupt enable for level 1 0 = OFF 1 = ON
26	0x0	MU1:MEM group up threshold interrupt enable for level 1 0 = OFF 1 = ON
25	0x0	MD0:MEM group down threshold interrupt enable for level 0 0 = OFF 1 = ON
24	0x0	MU0:MEM group up threshold interrupt enable for level 0 0 = OFF 1 = ON
23	0x0	GD3:GPU group down threshold interrupt enable for level 3 0 = OFF 1 = ON
22	0x0	GU3:GPU group up threshold interrupt enable for level 3 0 = OFF 1 = ON
21	0x0	GD2:GPU group down threshold interrupt enable for level 2 0 = OFF 1 = ON
20	0x0	GU2:GPU group up threshold interrupt enable for level 2 0 = OFF 1 = ON
19	0x0	GD1:GPU group down threshold interrupt enable for level 1 0 = OFF

Bit	Reset	Description
		1 = ON
18	0x0	GU1:GPU group up threshold interrupt enable for level 1 0 = OFF 1 = ON
17	0x0	GD0:GPU group down threshold interrupt enable for level 0 0 = OFF 1 = ON
16	0x0	GU0:GPU group up threshold interrupt enable for level 0 0 = OFF 1 = ON
15	0x0	CD3:CPU group down threshold interrupt enable for level 3 0 = OFF 1 = ON
14	0x0	CU3:CPU group up threshold interrupt enable for level 3 0 = OFF 1 = ON
13	0x0	CD2:CPU group down threshold interrupt enable for level 2 0 = OFF 1 = ON
12	0x0	CU2:CPU group up threshold interrupt enable for level 2 0 = OFF 1 = ON
11	0x0	CD1:CPU group down threshold interrupt enable for level 1 0 = OFF 1 = ON
10	0x0	CU1:CPU group up threshold interrupt enable for level 1 0 = OFF 1 = ON
9	0x0	CD0:CPU group down threshold interrupt enable for level 0 0 = OFF 1 = ON
8	0x0	CU0:CPU group up threshold interrupt enable for level 0 0 = OFF 1 = ON
7	0x0	TD3:TSENSE group down threshold interrupt enable for level 3 0 = OFF 1 = ON
6	0x0	TU3:TSENSE group up threshold interrupt enable for level 3 0 = OFF 1 = ON
5	0x0	TD2:TSENSE group down threshold interrupt enable for level 2 0 = OFF 1 = ON
4	0x0	TU2:TSENSE group up threshold interrupt enable for level 2 0 = OFF 1 = ON
3	0x0	TD1:TSENSE group down threshold interrupt enable for level 1 0 = OFF 1 = ON
2	0x0	TU1:TSENSE group up threshold interrupt enable for level 1 0 = OFF 1 = ON
1	0x0	TD0:TSENSE group down threshold interrupt enable for level 0 0 = OFF

Bit	Reset	Description
		1 = ON
0	0x0	TU0:TSENSE group up threshold interrupt enable for level 0 0 = OFF 1 = ON

37.4.28 SOC_THERM_THERMCTL_INTR_DIS_0

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

Bit	Reset	Description
31	0x0	MD3:MEM group down threshold interrupt disable for level 3 0 = OFF 1 = ON
30	0x0	MU3:MEM group up threshold interrupt disable for level 3 0 = OFF 1 = ON
29	0x0	MD2:MEM group down threshold interrupt disable for level 2 0 = OFF 1 = ON
28	0x0	MU2:MEM group up threshold interrupt disable for level 2 0 = OFF 1 = ON
27	0x0	MD1:MEM group down threshold interrupt disable for level 1 0 = OFF 1 = ON
26	0x0	MU1:MEM group up threshold interrupt disable for level 1 0 = OFF 1 = ON
25	0x0	MD0:MEM group down threshold interrupt disable for level 0 0 = OFF 1 = ON
24	0x0	MU0:MEM group up threshold interrupt disable for level 0 0 = OFF 1 = ON
23	0x0	GD3:GPU group down threshold interrupt disable for level 3 0 = OFF 1 = ON
22	0x0	GU3:GPU group up threshold interrupt disable for level 3 0 = OFF 1 = ON
21	0x0	GD2:GPU group down threshold interrupt disable for level 2 0 = OFF 1 = ON
20	0x0	GU2:GPU group up threshold interrupt disable for level 2 0 = OFF 1 = ON
19	0x0	GD1:GPU group down threshold interrupt disable for level 1 0 = OFF 1 = ON
18	0x0	GU1:GPU group up threshold interrupt disable for level 1 0 = OFF 1 = ON
17	0x0	GD0:GPU group down threshold interrupt disable for level 0 0 = OFF

Bit	Reset	Description
		1 = ON
16	0x0	GU0:GPU group up threshold interrupt disable for level 0 0 = OFF 1 = ON
15	0x0	CD3:CPU group down threshold interrupt disable for level 3 0 = OFF 1 = ON
14	0x0	CU3:CPU group up threshold interrupt disable for level 3 0 = OFF 1 = ON
13	0x0	CD2:CPU group down threshold interrupt disable for level 2 0 = OFF 1 = ON
12	0x0	CU2:CPU group up threshold interrupt disable for level 2 0 = OFF 1 = ON
11	0x0	CD1:CPU group down threshold interrupt disable for level 1 0 = OFF 1 = ON
10	0x0	CU1:CPU group up threshold interrupt disable for level 1 0 = OFF 1 = ON
9	0x0	CD0:CPU group down threshold interrupt disable for level 0 0 = OFF 1 = ON
8	0x0	CU0:CPU group up threshold interrupt disable for level 0 0 = OFF 1 = ON
7	0x0	TD3:TSENSE group down threshold interrupt disable for level 3 0 = OFF 1 = ON
6	0x0	TU3:TSENSE group up threshold interrupt disable for level 3 0 = OFF 1 = ON
5	0x0	TD2:TSENSE group down threshold interrupt disable for level 2 0 = OFF 1 = ON
4	0x0	TU2:TSENSE group up threshold interrupt disable for level 2 0 = OFF 1 = ON
3	0x0	TD1:TSENSE group down threshold interrupt disable for level 1 0 = OFF 1 = ON
2	0x0	TU1:TSENSE group up threshold interrupt disable for level 1 0 = OFF 1 = ON
1	0x0	TD0:TSENSE group down threshold interrupt disable for level 0 0 = OFF 1 = ON
0	0x0	TU0:TSENSE group up threshold interrupt disable for level 0 0 = OFF 1 = ON

37.4.29 SOC_THERM_THERMCTL_STATS_CTL_0

Offset: 0x94 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	0x0	CLEAR_DN: Clear DN statistics for all levels
2	0x0	ENB_DN: Enable DN statistics collection for all levels
1	0x0	CLEAR_UP: Clear statistics for all levels
0	0x0	ENB_UP: Enable statistics collection for all levels

37.4.30 SOC_THERM_THERMCTL_SLOWDOWN_THRESHOLD_0

Offset: 0x98 | Read/Write: R/W | Reset: 0x69696969 (0b01101001011010010110100101101001)

Bit	Reset	Description
31:24	0x69	MEM: Slowdown threshold for MEM group
23:16	0x69	GPU: Slowdown threshold for GPU group
15:8	0x69	CPU: Slowdown threshold for CPU group
7:0	0x69	TSENSE: Slowdown threshold for TSENSE group

37.4.31 SOC_THERM_THERMCTL_SLOWDOWN_CTL_0

Offset: 0x9c | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
31:30	0x0	SLOWDOWN_SELECT: Selects which one to throttle 0 = NONE 1 = CPU_ONLY 2 = GPU_ONLY 3 = BOTH
4	0x0	ANY_EN
3	0x0	MEM_EN
2	0x0	GPU_EN
1	0x0	CPU_EN
0	0x0	TSENSE_EN

37.4.32 SOC_THERM_TSENSOR_CPU0_CONFIG0_0

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: capture overflow

Bit	Reset	Description
		0 = OFF 1 = ON
1	0x0	RO_SEL: ring oscillator select 0 = TS 1 = VS
0	0x1	STOP: sensor stop 0 = OFF 1 = ON

37.4.33 SOC_THERM_TSENSOR_CPU0_CONFIG1_0

Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: Time between en and capture
20:15	0x0	TIDDQ_EN: Time between IDDQ and en
9:0	0x0	TSAMPLE: N count

37.4.34 SOC_THERM_TSENSOR_CPU0_CONFIG2_0

Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

37.4.35 SOC_THERM_TSENSOR_CPU0_STATUS0_0

Offset: 0xcc | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: valid capture available
15:0	X	CAPTURE:last valid raw capture

37.4.36 SOC_THERM_TSENSOR_CPU0_STATUS1_0

Offset: 0xd0 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: valid capture available
15:0	X	TEMP: last valid translated temperature

37.4.37 SOC_THERM_TSENSOR_CPU0_STATUS2_0

Offset: 0xd4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature

Bit	Reset	Description
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

37.4.38 SOC_THERM_TSENSOR_CPU1_CONFIG0_0

Offset: 0xe0 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: Add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: Capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: Ring oscillator select 0 = TS 1 = VS
0	0x1	STOP: Sensor stop 0 = OFF 1 = ON

37.4.39 SOC_THERM_TSENSOR_CPU1_CONFIG1_0

Offset: 0xe4 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: Time between en and capture
20:15	0x0	TIDDQ_EN: Time between IDDQ and en
9:0	0x0	TSAMPLE: N count

37.4.40 SOC_THERM_TSENSOR_CPU1_CONFIG2_0

Offset: 0xe8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

37.4.41 SOC_THERM_TSENSOR_CPU1_STATUS0_0

Offset: 0xec | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: Valid capture available

Bit	Reset	Description
15:0	X	CAPTURE: Last valid raw capture

37.4.42 SOC_THERM_TSENSOR_CPU1_STATUS1_0

Offset: 0xf0 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

37.4.43 SOC_THERM_TSENSOR_CPU1_STATUS2_0

Offset: 0xf4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

37.4.44 SOC_THERM_TSENSOR_CPU2_CONFIG0_0

Offset: 0x100 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: Add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: Capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: Ring oscillator select 0 = TS 1 = VS
0	0x1	STOP: Sensor stop 0 = OFF 1 = ON

37.4.45 SOC_THERM_TSENSOR_CPU2_CONFIG1_0

Offset: 0x104 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: Time between en and capture
20:15	0x0	TIDDQ_EN: Time between IDDQ and en

Bit	Reset	Description
9:0	0x0	TSAMPLE: N count

37.4.46 SOC_THERM_TSENSOR_CPU2_CONFIG2_0

Offset: 0x108 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

37.4.47 SOC_THERM_TSENSOR_CPU2_STATUS0_0

Offset: 0x10c | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID:valid capture available
15:0	X	CAPTURE:last valid raw capture

37.4.48 SOC_THERM_TSENSOR_CPU2_STATUS1_0

Offset: 0x110 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

37.4.49 SOC_THERM_TSENSOR_CPU2_STATUS2_0

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

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

37.4.50 SOC_THERM_TSENSOR_CPU3_CONFIG0_0

Offset: 0x120 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: Capture overflow 0 = OFF 1 = ON

Bit	Reset	Description
1	0x0	RO_SEL: Ring oscillator select 0 = TS 1 = VS
0	0x1	STOP: Sensor stop 0 = OFF 1 = ON

37.4.51 SOC_THERM_TSENSOR_CPU3_CONFIG1_0

Offset: 0x124 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: Time between en and capture
20:15	0x0	TIDDQ_EN: Time between IDDQ and en
9:0	0x0	TSAMPLE: N count

37.4.52 SOC_THERM_TSENSOR_CPU3_CONFIG2_0

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

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

37.4.53 SOC_THERM_TSENSOR_CPU3_STATUS0_0

Offset: 0x12c | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: Valid capture available
15:0	X	CAPTURE: Last valid raw capture

37.4.54 SOC_THERM_TSENSOR_CPU3_STATUS1_0

Offset: 0x130 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

37.4.55 SOC_THERM_TSENSOR_CPU3_STATUS2_0

Offset: 0x134 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

37.4.56 SOC_THERM_TSENSOR_MEM0_CONFIG0_0

Offset: 0x140 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: ring oscillator select 0 = TS 1 = VS
0	0x1	STOP: sensor stop 0 = OFF 1 = ON

37.4.57 SOC_THERM_TSENSOR_MEM0_CONFIG1_0

Offset: 0x144 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE:temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: time between en and capture
20:15	0x0	TIDDQ_EN: time between IDDQ and en
9:0	0x0	TSAMPLE: N count

37.4.58 SOC_THERM_TSENSOR_MEM0_CONFIG2_0

Offset: 0x148 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

37.4.59 SOC_THERM_TSENSOR_MEM0_STATUS0_0

Offset: 0x14c | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: Valid capture available
15:0	X	CAPTURE: Last valid raw capture

37.4.60 SOC_THERM_TSENSOR_MEM0_STATUS1_0

Offset: 0x150 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

37.4.61 SOC_THERM_TSENSOR_MEM0_STATUS2_0

Offset: 0x154 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

37.4.62 SOC_THERM_TSENSOR_MEM1_CONFIG0_0

Offset: 0x160 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: Add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: Capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: Ring oscillator select 0 = TS 1 = VS
0	0x1	STOP: Sensor stop 0 = OFF 1 = ON

37.4.63 SOC_THERM_TSENSOR_MEM1_CONFIG1_0

Offset: 0x164 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: Time between en and capture
20:15	0x0	TIDDQ_EN: Time between IDDQ and en
9:0	0x0	TSAMPLE: N count

37.4.64 SOC_THERM_TSENSOR_MEM1_CONFIG2_0

Offset: 0x168 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

37.4.65 SOC_THERM_TSENSOR_MEM1_STATUS0_0

Offset: 0x16c | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: Valid capture available
15:0	X	CAPTURE: Last valid raw capture

37.4.66 SOC_THERM_TSENSOR_MEM1_STATUS1_0

Offset: 0x170 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

37.4.67 SOC_THERM_TSENSOR_MEM1_STATUS2_0

Offset: 0x174 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

37.4.68 SOC_THERM_TSENSOR_GPU_CONFIG0_0

Offset: 0x180 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: Add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: Capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: Ring oscillator select 0 = TS 1 = VS
0	0x1	STOP: Sensor stop

Bit	Reset	Description
		0 = OFF 1 = ON

37.4.69 SOC_THERM_TSENSOR_GPU_CONFIG1_0

Offset: 0x184 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: Time between en and capture
20:15	0x0	TIDDQ_EN: Time between IDDQ and en
9:0	0x0	TSAMPLE: N count

37.4.70 SOC_THERM_TSENSOR_GPU_CONFIG2_0

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

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

37.4.71 SOC_THERM_TSENSOR_GPU_STATUS0_0

Offset: 0x18c | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: Valid capture available
15:0	X	CAPTURE: Last valid raw capture

37.4.72 SOC_THERM_TSENSOR_GPU_STATUS1_0

Offset: 0x190 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

37.4.73 SOC_THERM_TSENSOR_GPU_STATUS2_0

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

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

37.4.74 SOC_THERM_TSENSOR_PLLX_CONFIG0_0

Offset: 0x1a0 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: Add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: Capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: Ring oscillator select 0 = TS 1 = VS
0	0x1	STOP: Sensor stop 0 = OFF 1 = ON

37.4.75 SOC_THERM_TSENSOR_PLLX_CONFIG1_0

Offset: 0x1a4 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: time between en and capture
20:15	0x0	TIDDQ_EN: time between IDDQ and en
9:0	0x0	TSAMPLE: N count

37.4.76 SOC_THERM_TSENSOR_PLLX_CONFIG2_0

Offset: 0x1a8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

37.4.77 SOC_THERM_TSENSOR_PLLX_STATUS0_0

Offset: 0x1ac | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: Valid capture available
15:0	X	CAPTURE: Last valid raw capture

37.4.78 SOC_THERM_TSENSOR_PLLX_STATUS1_0

Offset: 0x1b0 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

37.4.79 SOC_THERM_TSENSOR_PLLX_STATUS2_0

Offset: 0x1b4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

37.4.80 SOC_THERM_TSENSOR_PDIV_0

Offset: 0x1c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:12	0x0	CPU_PDIV: PDIV for TS_CPU0 TS_CPU3
11:8	0x0	GPU_PDIV: PDIV for TS_GPU
7:4	0x0	MEM_PDIV: PDIV for TS_MEM
3:0	0x0	PLLX_PDIV: PDIV for TS_PLLX

37.4.81 SOC_THERM_TSENSOR_HOTSPOT_OFF_0

Offset: 0x1c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:16	0x0	CPU_HOTSPOT_OFF:CPU hotspot offset from PLLX
15:8	0x0	GPU_HOTSPOT_OFF:GPU hotspot offset from PLLX
7:0	0x0	MEM_HOTSPOT_OFF:MEM hotspot offset from PLLX

37.4.82 SOC_THERM_TSENSOR_TEMP1_0

Offset: 0x1c8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000))

Bit	Reset	Description
31:16	0x0	CPU_TEMP: Processed CPU temperature seen by thermal throttling logic. Temp readback format
15:0	0x0	GPU_TEMP: Processed GPU temperature seen by thermal throttling logic. Temp readback format

37.4.83 SOC_THERM_TSENSOR_TEMP2_0

Offset: 0x1cc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000))

Bit	Reset	Description
31:16	0x0	MEM_TEMP: Processed MEM temperature seen by thermal throttling logic. Temperature readback format

Bit	Reset	Description
15:0	0x0	SENSOR_TEMP: Processed sensor (PLLX) temperature seen by thermal throttling logic. Temperature readback format

37.4.84 SOC_THERM_TSENSOR_TSENSOR_PWR_VLD_OVERRIDE_0

Offset: 0x1d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	INVALIDATE_ON_PWR_GATING: Use CPU/GPU virtual power gating status to invalidate the CPU/GPU sensors if set otherwise use Rail gating status

37.4.85 SOC_THERM_TSENSOR_SPARE_ECO_0

Offset: 0x1d4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	Reserved.

37.4.86 SOC_THERM_TSENSOR_TEMP_SW_OVERRIDE_0

Offset: 0x1d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	SW_OVERRIDE_EN: 1: Enable software override of TSENSOR TEMP registers.

37.4.87 SOC_THERM_EDP_OC_ALARM_OC1_CFG_0

Offset: 0x310 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0000100)

Bit	Reset	Description
6	0x0	LONG_LATENCY_THROTTLE: Used to inform the CAR PLLx hardware controller that hardware is expected to control the PLL for large amounts of time when this OC alarm is engaged
5	0x0	HW_RESTORE_EN: Auto-restore system to unthrottled state after this alarm is deasserted 0 = DISABLE 1 = ENABLE
4	0x0	PWRGOOD_MASK: Mask the throttling if power is not good. 0 = ENABLE_THROTTLE 1 = DISABLE_THROTTLE
3:2	0x1	THROTTLE_MODE: To select sticky versus brief mode throttling 0 = DISABLED 1 = STICKY 2 = BRIEF 3 = RESERVED
1	0x0	ALARM_POLARITY:1-assert low, 0-assert high
0	0x0	EN_THROTTLE: Enable the throttling on this over current alarm

37.4.88 SOC_THERM_EDP_OC_ALARM_OC1_CNT_THRESHOLD_0

Offset: 0x314 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	EVENT_CNT_THRESHOLD: Event count threshold to raise interrupt, 0-rise interrupt for every event

37.4.89 SOC_THERM_EDP_OC_ALARM_OC1_THROTTLE_PERIOD_0

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

Bit	Reset	Description
31:0	0x0	OC_THROTTLE_PERIOD: Brief throttle period in microseconds. Valid only for brief throttle

37.4.90 SOC_THERM_EDP_OC_ALARM_OC1_COUNT_0

Hardware uses the COUNT register to create a throttling event (by comparing with threshold). This register is reset every time the FILTER timer expires.

Offset: 0x31c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Event count

37.4.91 SOC_THERM_EDP_OC_ALARM_OC1_FILTER_0

Offset: 0x320 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	COUNT: Event count

37.4.92 SOC_THERM_EDP_OC_ALARM_OC2_CFG_0

Offset: 0x324 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0000100)

Bit	Reset	Description
6	0x0	LONG_LATENCY_THROTTLE: Used to inform the CAR PLLx hardware controller that hardware is expected to control the PLL for large amounts of time when this OC alarm is engaged
5	0x0	HW_RESTORE_EN: Auto-restore system to unthrottled state after this alarm is deasserted 0 = DISABLE 1 = ENABLE
4	0x0	PWRGOOD_MASK: Mask the throttling if power is not good. 0 = ENABLE_THROTTLE 1 = DISABLE_THROTTLE
3:2	0x1	THROTTLE_MODE: To select sticky versus brief mode throttling 0 = DISABLED 1 = STICKY 2 = BRIEF 3 = RESERVED
1	0x0	ALARM_POLARITY:1-assert low, 0-assert high
0	0x0	EN_THROTTLE: Enable the throttling on this over current alarm

37.4.93 SOC_THERM_EDP_OC_ALARM_OC2_CNT_THRESHOLD_0

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

Bit	Reset	Description
31:0	0x0	EVENT_CNT_THRESHOLD: Event count threshold to raise interrupt, 0-rise interrupt for every event

37.4.94 SOC_THERM_EDP_OC_ALARM_OC2_THROTTLE_PERIOD_0

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

Bit	Reset	Description
31:0	0x0	OC_THROTTLE_PERIOD: Brief throttle period in microseconds. Valid only for brief throttle

37.4.95 SOC_THERM_EDP_OC_ALARM_OC2_COUNT_0

Hardware uses the COUNT register to create a throttling event (by comparing with threshold). This register is reset every time the FILTER timer expires.

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

Bit	Reset	Description
31:0	X	COUNT: Event count

37.4.96 SOC_THERM_EDP_OC_ALARM_OC2_FILTER_0

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

Bit	Reset	Description
31:0	0x0	COUNT: Event count

37.4.97 SOC_THERM_EDP_OC_ALARM_OC3_CFG_0

Offset: 0x338 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0000100)

Bit	Reset	Description
6	0x0	LONG_LATENCY_THROTTLE: Used to inform the CAR PLLx hardware controller that hardware is expected to control the PLL for large amounts of time when this OC alarm is engaged
5	0x0	HW_RESTORE_EN: Auto-restore system to unthrottled state after this alarm is deasserted 0 = DISABLE 1 = ENABLE
4	0x0	PWRGOOD_MASK: Mask the throttling if power is not good. 0 = ENABLE_THROTTLE 1 = DISABLE_THROTTLE
3:2	0x1	THROTTLE_MODE: To select sticky versus brief mode throttling 0 = DISABLED 1 = STICKY 2 = BRIEF 3 = RESERVED
1	0x0	ALARM_POLARITY:1-assert low, 0-assert high
0	0x0	EN_THROTTLE: Enable the throttling on this over current alarm

37.4.98 SOC_THERM_EDP_OC_ALARM_OC3_CNT_THRESHOLD_0

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

Bit	Reset	Description
31:0	0x0	EVENT_CNT_THRESHOLD: Event count threshold to raise interrupt, 0-rise interrupt for every event

37.4.99 SOC_THERM_EDP_OC_ALARM_OC3_THROTTLE_PERIOD_0

Offset: 0x340 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	OC_THROTTLE_PERIOD: Brief throttle period in microseconds- valid only for brief throttle

37.4.100 SOC_THERM_EDP_OC_ALARM_OC3_COUNT_0

Hardware uses the COUNT register to create a throttling event (by comparing with threshold). This register is reset every time the FILTER timer expires.

Offset: 0x344 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	COUNT: Event count

37.4.101 SOC_THERM_EDP_OC_ALARM_OC3_FILTER_0

Offset: 0x348 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	COUNT: Event count

37.4.102 SOC_THERM_EDP_OC_ALARM_OC4_CFG_0

Offset: 0x34c | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0000100)

Bit	Reset	Description
6	0x0	LONG_LATENCY_THROTTLE: Used to inform the CAR PLLx hardware controller that hardware is expected to control the PLL for large amounts of time when this OC alarm is engaged
5	0x0	HW_RESTORE_EN: Auto-restore system to unthrottled state after this alarm is deasserted 0 = DISABLE 1 = ENABLE
4	0x0	PWRGOOD_MASK: Mask the throttling if power is not good. 0 = ENABLE_THROTTLE 1 = DISABLE_THROTTLE
3:2	0x1	THROTTLE_MODE: To select sticky versus brief mode throttling 0 = DISABLED 1 = STICKY 2 = BRIEF 3 = RESERVED
1	0x0	ALARM_POLARITY:1-assert low, 0-assert high
0	0x0	EN_THROTTLE: Enable the throttling on this over current alarm

37.4.103 SOC_THERM_EDP_OC_ALARM_OC4_CNT_THRESHOLD_0

Offset: 0x350 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	EVENT_CNT_THRESHOLD: Event count threshold to raise interrupt, 0-rise interrupt for every event

37.4.104 SOC_THERM_EDP_OC_ALARM_OC4_THROTTLE_PERIOD_0

Offset: 0x354 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	OC_THROTTLE_PERIOD: Brief throttle period in microseconds- valid only for brief throttle

37.4.105 SOC_THERM_EDP_OC_ALARM_OC4_COUNT_0

Hardware uses the COUNT register to create a throttling event (by comparing with threshold). This register is reset every time the FILTER timer expires.

Offset: 0x358 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Event count

37.4.106 SOC_THERM_EDP_OC_ALARM_OC4_FILTER_0

Offset: 0x35c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	COUNT: Event count

37.4.107 SOC_THERM_EDP_OC_ALARM_OC5_CFG_0

Offset: 0x360 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0000100)

This register is reserved.

37.4.108 SOC_THERM_EDP_OC_ALARM_OC5_CNT_THRESHOLD_0

Offset: 0x364 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

This register is reserved.

37.4.109 SOC_THERM_EDP_OC_ALARM_OC5_THROTTLE_PERIOD_0

Offset: 0x368 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

This register is reserved.

37.4.110 SOC_THERM_EDP_OC_ALARM_OC5_COUNT_0

Offset: 0x36c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

This register is reserved.

37.4.111 SOC_THERM_EDP_OC_ALARM_OC5_FILTER_0

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

This register is reserved.

37.4.112 SOC_THERM_EDP_OC_INTR_STATUS_0

Offset: 0x39c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx0xx00xxxxxxxxxx00000)

Bit	Reset	Description
20	0x0	CLDVFS_DIDT:CLDVFS interrupt status
4	0x0	OC5: Reserved.
3	0x0	OC4: OC event 4 interrupt status
2	0x0	OC3: OC event 3 interrupt status
1	0x0	OC2: OC event 2 interrupt status
0	0x0	OC1: OC event 1 interrupt status

37.4.113 SOC_THERM_EDP_OC_INTR_ENABLE_0

Offset: 0x3a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx0xx00xxxxxxxxxx00000)

Bit	Reset	Description
20	0x0	CLDVFS_DIDT_EN:CLDVFS interrupt enable
4	0x0	OC5_EN: OC Reserved
3	0x0	OC4_EN: OC event 4 interrupt enable
2	0x0	OC3_EN: OC event 3 interrupt enable
1	0x0	OC2_EN: OC event 2 interrupt enable
0	0x0	OC1_EN: OC event 1 interrupt enable

37.4.114 SOC_THERM_EDP_OC_INTR_DISABLE_0

Offset: 0x3a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx0xx00xxxxxxxxxx00000)

Bit	Reset	Description
20	0x0	CLDVFS_DIDT_DIS:CLDVFS interrupt disable
4	0x0	OC5_DIS: Reserved
3	0x0	OC4_DIS: OC event 4 interrupt disable
2	0x0	OC3_DIS: OC event 3 interrupt disable
1	0x0	OC2_DIS: OC event 2 interrupt disable
0	0x0	OC1_DIS: OC event 1 interrupt disable

37.4.115 SOC_THERM_EDP_OC_ALARM_OC1_STATS_0

This register counts the number of overcurrent (OC) occurrences until it is cleared by software.

Offset: 0x3a8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	EVENT_COUNT: Statistics for event count

37.4.116 SOC_THERM_EDP_OC_ALARM_OC2_STATS_0

This register counts the number of overcurrent (OC) occurrences until it is cleared by software.

Offset: 0x3ac │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	EVENT_COUNT: Statistics for event count

37.4.117 SOC_THERM_EDP_OC_ALARM_OC3_STATS_0

This register counts the number of overcurrent (OC) occurrences until it is cleared by software.

Offset: 0x3b0 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	EVENT_COUNT: Statistics for event count

37.4.118 SOC_THERM_EDP_OC_ALARM_OC4_STATS_0

This register counts the number of overcurrent (OC) occurrences until it is cleared by software.

Offset: 0x3b4 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	EVENT_COUNT: Statistics for event count

37.4.119 SOC_THERM_EDP_OC_ALARM_OC5_STATS_0

Offset: 0x3b8 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

This register is reserved.

37.4.120 SOC_THERM_EDP_OC_ALARM_STATS_CTRL_0

Offset: 0x3c4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	CLEAR_ALL: Clear all statistics
0	0x0	ENB_ALL: Enable all statistics collections for all counters

37.4.121 SOC_THERM_EDP_OC_CLDVFS_DIDT_CNT_THRESHOLD_0

Offset: 0x3c8 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	EVENT_CNT_THRESHOLD: Event count threshold to raise interrupt, 0-rise interrupt for every event

37.4.122 SOC_THERM_EDP_OC_CLDVFS_DIDT_EVENT_COUNT_0

Offset: 0x3cc │ Read/Write: RO │ Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	COUNT: Event count

37.4.123 SOC_THERM_EDP_OC_CLDVFS_DIDT_EVENT_CNT_FILTER_0

Offset: 0x3d0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	COUNT: event count

37.4.124 SOC_THERM_EDP_OC_CLDVFS_DIDT_EVENT_STATS_0

Offset: 0x3d4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	EVENT_COUNT: Statistics for event count

37.4.125 SOC_THERM_EDP_OC_ALARM_THROTTLE_PERIOD_CTL_0

Offset: 0x3d8 | Read/Write: R/W | Reset: 0x000000cc (0bxxxxxxxxxxxxxxxxxxxxxxxx11001100))

Bit	Reset	Description
7:0	0xcc	NUM_CLKS_IN_1US: Number of soc_therm clocks in 1µs used to determine brief throttle length (default based on 204 MHz)

37.4.126 SOC_THERM_THROTTLECTL_GLOBAL_THROTTLE_CFG_0

Offset: 0x400 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0001)

Bit	Reset	Description
3	0x0	DFLL_PSKIP_CTRL:0: (default) pause CPU pulse skipper when cldvfs2soc_therm_skipper1_en is asserted1: do not pause CPU pulse skipper
2	0x0	PSKIP_RESTORE_CTRL:0: Software does not restore (default) 1: Software restores. If HW_RESTORE_EN==0, hardware will not restore the pulse skipper
1	0x0	SW_OVERRIDE_MODE: Pulse skipper software override :=1 causes all throttle indicators to use software pulse skipper cfg for throttling pulse skipper 0 = DISABLE 1 = ENABLE
0	0x1	ENB: Single bit to disable all throttling 0 = DISABLE 1 = ENABLE

37.4.127 SOC_THERM_THROTTLECTL_SW_CPU_PSKIP_CTRL_0

Offset: 0x408 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

37.4.128 SOC_THERM_THROTTLECTL_SW_CPU_PSKIP_RAMP_RATE_0

Offset: 0x40c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

37.4.129 SOC_THERM_THROTTLECTL_SW_GPU_PSKIP_CTRL_0

Offset: 0x410 | Read/Write: R/W | Reset: 0xX0000000 (0b0xxxxxxxxxxxx00000000000000000)

Bit	Reset	R/W	Description
31	0x0	RW	Reserved
30	0x1	RO	LOW_MED_HIGH: This field is set to 1'b1 by hardware after reset. Software needs to read this field to decide the style of GPU throttling. When this bit is read back as 1'b1, this means the LOW/MEDIUM/HIGH interface is used. When this bit is read back as 1'b0, it means the legacy GPU throttling with PSKIP is used.
18:16	0x0	RW	THROTTLE_DEPTH: Configures LOW/MEDIUM/HIGH throttling (zero-one-hot encoding).
15:8	0x0	RW	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	RW	SUPER_CDIV_DIVISOR: Actual Value = n+1

37.4.130 SOC_THERM_THROTTLECTL_SW_GPU_PSKIP_RAMP_RATE_0

Offset: 0x414 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLED 1 = ENABLED
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

37.4.131 SOC_THERM_THROTTLECTL_CPU_PSKIP_STATUS_0

Offset: 0x418 | Read/Write: RO | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:12	X	CURR_PULSE_SKIP_M:
11:4	X	CURR_PULSE_SKIP_N:
1	X	SW_OVERRIDE_STATUS: 0 = DISABLED 1 = ENABLED
0	X	ENB_STATUS: 0 = DISABLED 1 = ENABLED

37.4.132 SOC_THERM_THROTTLECTL_GPU_PSKIP_STATUS_0

Offset: 0x41c | Read/Write: RO | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:12	X	CURR_PULSE_SKIP_M:
11:4	X	CURR_PULSE_SKIP_N:
1	X	SW_OVERRIDE_STATUS: 0 = DISABLED 1 = ENABLED
0	X	ENB_STATUS: 0 = DISABLED 1 = ENABLED

37.4.133 SOC_THERM_THROTTLECTL_PRIORITY_LOCK_0

Offset: 0x424 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Maximum priority allowed for software programmable vectors

37.4.134 SOC_THERM_THROTTLECTL_THROTTLE_STATUS_0

Offset: 0x428 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
12	X	PRIORITY_LOCK_BREACH: Set to '1' if the selected throttle vector is of higher priority than the limit specified by the boot loader. The field is cleared when software updates PRIORITY registers.
11:4	X	THROTTLE_SEQ_STATE:0=WAIT
0	X	ENB_STATUS: Global enable status 0 = DISABLED 1 = ENABLED

37.4.135 SOC_THERM_THROTTLECTL_LITE_CPU_PSKIP_CTRL_0

Offset: 0x430 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

37.4.136 SOC_THERM_THROTTLECTL_LITE_CPU_PSKIP_RAMP_RATE_0

Offset: 0x434 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

37.4.137 SOC_THERM_THROTTLECTL_LITE_GPU_PSKIP_CTRL_0

Offset: 0x438 | Read/Write: R/W | Reset: 0xX0000000 (0b0xxxxxxxxxxxx00000000000000000000)

Bit	Reset	R/W	Description
31	0x0	RW	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
30	X	RO	LOW_MED_HIGH: Throttling configured through THROTTLE_DEPTH 0 = DISABLE 1 = ENABLE
18:16	0x0	RW	THROTTLE_DEPTH: Configures LOW/MEDIUM/HIGH throttling (zero-one-hot encoding)
15:8	0x0	RW	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	RW	SUPER_CDIV_DIVISOR: Actual Value = n+1

37.4.138 SOC_THERM_THROTTLECTL_LITE_GPU_PSKIP_RAMP_RATE_0

Offset: 0x43c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

37.4.139 SOC_THERM_THROTTLECTL_LITE_THROTTLE_PRIORITY_0

Offset: 0x444 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Bigger value indicates higher priority. In general, higher priority translates to lower target frequency

37.4.140 SOC_THERM_THROTTLECTL_LITE_THROTTLE_DELAY_0

Offset: 0x448 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DELAY_VAL: Number of soc_therm_clk cycles to delay the alert

37.4.141 SOC_THERM_THROTTLECTL_HEAVY_CPU_PSKIP_CTRL_0

Offset: 0x460 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

37.4.142 SOC_THERM_THROTTLECTL_HEAVY_CPU_PSKIP_RAMP_RATE_0

Offset: 0x464 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

37.4.143 SOC_THERM_THROTTLECTL_HEAVY_GPU_PSKIP_CTRL_0

Offset: 0x468 | Read/Write: R/W | Reset: 0xX0000000 (0b0xxxxxxxxxxxx00000000000000000000))

Bit	Reset	R/W	Description
31	0x0	RW	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
30	X	RO	LOW_MED_HIGH: Throttling configured through THROTTLE_DEPTH 0 = DISABLE 1 = ENABLE
18:16	0x0	RW	THROTTLE_DEPTH: Configures LOW/MEDIUM/HIGH throttling (zero-one-hot encoding)
15:8	0x0	RW	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	RW	SUPER_CDIV_DIVISOR: Actual Value = n+1

37.4.144 SOC_THERM_THROTTLECTL_HEAVY_GPU_PSKIP_RAMP_RATE_0

Offset: 0x46c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

37.4.145 SOC_THERM_THROTTLECTL_HEAVY_THROTTLE_PRIORITY_0

Offset: 0x474 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Bigger value indicates higher priority. In general, higher priority translates to lower target frequency

37.4.146 SOC_THERM_THROTTLECTL_HEAVY_THROTTLE_DELAY_0

Offset: 0x478 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DELAY_VAL: Number of soc_therm_clk cycles to delay the alert

37.4.147 SOC_THERM_THROTTLECTL_OC1_CPU_PSKIP_CTRL_0

Offset: 0x490 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

37.4.148 SOC_THERM_THROTTLECTL_OC1_CPU_PSKIP_RAMP_RATE_0

Offset: 0x494 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

37.4.149 SOC_THERM_THROTTLECTL_OC1_GPU_PSKIP_CTRL_0

Offset: 0x498 | Read/Write: R/W | Reset: 0xX0000000 (0b0xxxxxxxxxxxx000000000000000000))

Bit	Reset	R/W	Description
31	0x0	RW	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
30	X	RO	LOW_MED_HIGH: Throttling configured through THROTTLE_DEPTH 0 = DISABLE 1 = ENABLE
18:16	0x0	RW	THROTTLE_DEPTH: Configures LOW/MEDIUM/HIGH throttling (zero-one-hot encoding)
15:8	0x0	RW	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	RW	SUPER_CDIV_DIVISOR: Actual Value = n+1

37.4.150 SOC_THERM_THROTTLECTL_OC1_GPU_PSKIP_RAMP_RATE_0

Offset: 0x49c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

37.4.151 SOC_THERM_THROTTLECTL_OC1_THROTTLE_PRIORITY_0

Offset: 0x4a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Bigger value indicates higher priority. In general, higher priority translates to lower target

Bit	Reset	Description
		frequency

37.4.152 SOC_THERM_THROTTLECTL_OC1_THROTTLE_DELAY_0

Offset: 0x4a8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DELAY_VAL: Number of soc_therm_clk cycles to delay the alert

37.4.153 SOC_THERM_THROTTLECTL_OC2_CPU_PSKIP_CTRL_0

Offset: 0x4c0 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

37.4.154 SOC_THERM_THROTTLECTL_OC2_CPU_PSKIP_RAMP_RATE_0

Offset: 0x4c4 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

37.4.155 SOC_THERM_THROTTLECTL_OC2_GPU_PSKIP_CTRL_0

Offset: 0x4c8 | Read/Write: R/W | Reset: 0xX0000000 (0b0xxxxxxxxxxxx00000000000000000000))

Bit	Reset	R/W	Description
31	0x0	RW	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
30	X	RO	LOW_MED_HIGH:Throttling configured through THROTTLE_DEPTH 0 = DISABLE 1 = ENABLE
18:16	0x0	RW	THROTTLE_DEPTH: Configures LOW/MEDIUM/HIGH throttling (zero-one-hot encoding)
15:8	0x0	RW	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	RW	SUPER_CDIV_DIVISOR: Actual Value = n+1

37.4.156 SOC_THERM_THROTTLECTL_OC2_GPU_PSKIP_RAMP_RATE_0

Offset: 0x4cc | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

37.4.157 SOC_THERM_THROTTLECTL_OC2_THROTTLE_PRIORITY_0

Offset: 0x4d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Bigger value indicates higher priority. In general, higher priority translates to lower target frequency

37.4.158 SOC_THERM_THROTTLECTL_OC2_THROTTLE_DELAY_0

Offset: 0x4d8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DELAY_VAL: Number of soc_therm_clk cycles to delay the alert

37.4.159 SOC_THERM_THROTTLECTL_OC3_CPU_PSKIP_CTRL_0

Offset: 0x4f0 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

37.4.160 SOC_THERM_THROTTLECTL_OC3_CPU_PSKIP_RAMP_RATE_0

Offset: 0x4f4 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

37.4.161 SOC_THERM_THROTTLECTL_OC3_GPU_PSKIP_CTRL_0

Offset: 0x4f8 | Read/Write: R/W | Reset: 0xX0000000 (0b0xxxxxxxxxxxx00000000000000000000))

Bit	Reset	R/W	Description
31	0x0	RW	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
30	X	RO	LOW_MED_HIGH: Throttling configured through THROTTLE_DEPTH 0 = DISABLE

Bit	Reset	R/W	Description
			1 = ENABLE
18:16	0x0	RW	THROTTLE_DEPTH: Configures LOW/MEDIUM/HIGH throttling (zero-one-hot encoding)
15:8	0x0	RW	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	RW	SUPER_CDIV_DIVISOR: Actual Value = n+1

37.4.162 SOC_THERM_THROTTLECTL_OC3_GPU_PSKIP_RAMP_RATE_0

Offset: 0x4fc | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

37.4.163 SOC_THERM_THROTTLECTL_OC3_THROTTLE_PRIORITY_0

Offset: 0x504 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Bigger value indicates higher priority. In general, higher priority translates to lower target frequency

37.4.164 SOC_THERM_THROTTLECTL_OC3_THROTTLE_DELAY_0

Offset: 0x508 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DELAY_VAL: Number of soc_therm_clk cycles to delay the alert

37.4.165 SOC_THERM_THROTTLECTL_OC4_CPU_PSKIP_CTRL_0

Offset: 0x520 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

37.4.166 SOC_THERM_THROTTLECTL_OC4_CPU_PSKIP_RAMP_RATE_0

Offset: 0x524 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

37.4.167 SOC_THERM_THROTTLECTL_OC4_GPU_PSKIP_CTRL_0

Offset: 0x528 | Read/Write: R/W | Reset: 0xX0000000 (0b0xxxxxxxxxxxx00000000000000000000))

Bit	Reset	R/W	Description
31	0x0	RW	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
30	X	RO	LOW_MED_HIGH: Throttling configured through THROTTLE_DEPTH 0 = DISABLE 1 = ENABLE
18:16	0x0	RW	THROTTLE_DEPTH: Configures LOW/MEDIUM/HIGH throttling (zero-one-hot encoding)
15:8	0x0	RW	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	RW	SUPER_CDIV_DIVISOR: Actual Value = n+1

37.4.168 SOC_THERM_THROTTLECTL_OC4_GPU_PSKIP_RAMP_RATE_0

Offset: 0x52c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

37.4.169 SOC_THERM_THROTTLECTL_OC4_THROTTLE_PRIORITY_0

Offset: 0x534 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Bigger value indicates higher priority. In general, higher priority translates to lower target frequency

37.4.170 SOC_THERM_THROTTLECTL_OC4_THROTTLE_DELAY_0

Offset: 0x538 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DELAY_VAL: Number of soc_therm_clk cycles to delay the alert

37.4.171 SOC_THERM_THROTTLECTL_OC5_CPU_PSKIP_CTRL_0

Offset: 0x550 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

This register is reserved.

37.4.172 SOC_THERM_THROTTLECTL_OC5_CPU_PSKIP_RAMP_RATE_0

Offset: 0x554 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

This register is reserved.

37.4.173 SOC_THERM_THROTTLECTL_OC5_GPU_PSKIP_CTRL_0

Offset: 0x558 | Read/Write: R/W | Reset: 0xX0000000 (0b0xxxxxxxxxxxx00000000000000000)

This register is reserved.

37.4.174 SOC_THERM_THROTTLECTL_OC5_GPU_PSKIP_RAMP_RATE_0

Offset: 0x55c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

This register is reserved.

37.4.175 SOC_THERM_THROTTLECTL_OC5_THROTTLE_PRIORITY_0

Offset: 0x564 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

This register is reserved.

37.5 Temperature Sensor Calibration Registers

37.5.1 FUSE_TSENSOR1_CALIB_0

Chip Option: tsensor1_calib

NVJTAG - temperature sensor calibration

Write protected by (production_mode & ~jtag_controls_fuse_sync) | security_mode

Offset: 0x184 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	TSENSOR1_CALIB

37.5.2 FUSE_TSENSOR2_CALIB_0

Chip Option: tsensor2_calib

NVJTAG - temperature sensor calibration

Write protected by (production_mode & ~jtag_controls_fuse_sync) | security_mode

Offset: 0x188 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	TSENSOR2_CALIB

37.5.3 FUSE_TSENSOR0_CALIB_0

Chip Option: tsensor0_calib

NVJTAG - temperature sensor calibration CP_TS_CPU1_offset=tsensor0_calib[12:0], FT_TS_CPU1_offset=tsensor0_calib[25:13]

Write protected by (production_mode & ~jtag_controls_fuse_sync) | security_mode

Offset: 0x198 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	TSENSOR0_CALIB

37.5.4 FUSE_TSENSOR3_CALIB_0

Chip Option: tsensor3_calib

NVJTAG - temperature sensor calibration

Write protected by (production_mode & ~jtag_controls_fuse_sync) | security_mode

Offset: 0x22c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	TSENSOR3_CALIB

37.5.5 FUSE_TSENSOR4_CALIB_0

Chip Option: tsensor4_calib

NVJTAG - temperature sensor calibration

Write protected by (production_mode & ~jtag_controls_fuse_sync) | security_mode

Offset: 0x254 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	TSENSOR4_CALIB

37.5.6 FUSE_TSENSOR5_CALIB_0

Chip Option: tsensor5_calib

NVJTAG - temperature sensor calibration

Write protected by (production_mode & ~jtag_controls_fuse_sync) | security_mode

Offset: 0x258 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	TSENSOR5_CALIB

37.5.7 FUSE_TSENSOR6_CALIB_0

Chip Option: tsensor6_calib

NVJTAG - temperature sensor calibration

Write protected by (production_mode & ~jtag_controls_fuse_sync) | security_mode

Offset: 0x25c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	TSENSOR6_CALIB

37.5.8 FUSE_TSENSOR7_CALIB_0

Chip Option: tsensor7_calib

NVJTAG - temperature sensor calibration

Write protected by (production_mode & ~jtag_controls_fuse_sync) | security_mode

Offset: 0x260 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	TSENSOR7_CALIB

37.5.9 FUSE_TSENSOR8_CALIB_0

Chip Option: tsensor8_calib

NVJTAG - temperature sensor for thermtrip

Write protected by (production_mode & ~jtag_controls_fuse_sync) | security_mode

Offset: 0x280 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	TSENSOR8_CALIB

37.5.10 FUSE_SPARE_REALIGNMENT_REG_0

Chip Option: spare_realignment_reg

NVJTAG - temperature sensor calibration

Write protected by (production_mode & ~jtag_controls_fuse_sync) | security_mode

Offset: 0x2fc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx00000000000000)

Bit	Reset	Description
13:0	0x0	SPARE_REALIGNMENT_REG

38.0 AUDIO-VIDEO PROCESSOR (AVP)

38.1 Overview

The AVP, or Audio-Video Processor, is an ARM7™ based processor sub-system used to:

- Manage the initial stages of boot
- Control and assist the hardware audio decoding blocks, BSEA and VCP2
- Control and assist the hardware video decoder, VDE

It can control the system and has access to the entire memory map (except for the PCIe® controller, GPU aperture, and CPU internal hardware).

The AVP has 256 Kbytes of local RAM, known as IRAM, for local code and data. IRAM is very low latency to the AVP, typically single cycle, and is also low power. Critical code and data should be stored in IRAM for the best performance and lowest power.

The AVP has access to DRAM either through an 8KB unified cache, or through an uncached memory aperture.

38.2 Address Map

Refer to the *Address Map* section of this document for the address map details.

38.3 Cache Controller

Refer to the *AVP Cache Controller* section of this document for the details of the cache controller.

38.4 Interrupts

Interrupts are controlled for the AVP by the system interrupt controller. Refer to the *Interrupt Controller* section of this document. All system interrupts are available to be routed to the AVP.

38.5 IRAM and Crossbar Bus

IRAM is configured as four contiguous banks of 64 Kilobytes of RAM. Each of these banks is accessible over a crossbar bus architecture, so that simultaneous access is possible to each bank from the crossbar masters.

The masters on the crossbar bus are:

- The ARM7 core
- The CPU complex over the AXI to ARM7 bridge
- The VCP2 audio acceleration block
- The UCQ interface of the VDE and BSEA
- The AHB bus interface

38.6 AVP Registers

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

Arbitration for the various resources on the crossbar bus can be configured via the Arbitration Priority registers. Each crossbar master can be assigned a priority for each crossbar resource so that the resource controller can evaluate the latency needs for each master and complete the requests in the most efficient manner.

It is important that each master is assigned a different priority for each resource. If any two priorities are the same the hardware will override the settings with "safe" values, which may not be what the programmer has intended. So when updating the priorities, you should program all of the priority registers at the same time (one for each master).

38.6.1 ARB_PRIO_CPU_PRIORITY_0

Shared Resource Priority for CPU Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x08040090 (0bxx001000000001xxx000000010010000)

Bit	Reset	Description
29:27	0x1	VDE: 000=Highest Priority 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
26:24	0x0	HOST1X: 000=Highest Priority 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
23:21	0x0	APB: Access Privilege to APB Bus Registers 00=Highest Priority 11=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = LOWEST
20:18	0x1	PSB: Access Privilege to PSB Registers 00=Highest Priority 11=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = LOWEST
14:12	0x0	IROM0: 000=Highest Priority 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
11:9	0x0	IRAM3: 000=Highest Priority 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6

Bit	Reset	Description
		7 = LOWEST
8:6	0x2	IRAM2: 000=Highest Priority 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
5:3	0x2	IRAM1: 000=Highest Priority 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
2:0	0x0	IRAM0: 000=Highest Priority 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST

38.6.2 ARB_PRIO_COP_PRIORITY_0

Shared Resource Priority for COP Register

Offset: 0x4 | Read/Write: R/W | Reset: 0x012024c2 (0bxx000001001000xxx010010011000010)

Bit	Reset	Description
29:27	0x0	VDE: 000=Highest Priority 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
26:24	0x1	HOST1X: 000=Highest Priority 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
23:21	0x1	APB: 11=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = LOWEST

Bit	Reset	Description
20:18	0x0	PSB: 11=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = LOWEST
14:12	0x2	IROM0: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
11:9	0x2	IRAM3: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
8:6	0x3	IRAM2: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
5:3	0x0	IRAM1: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
2:0	0x2	IRAM0: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST

38.6.3 ARB_PRIO_VCP_PRIORITY_0

Shared Resource Priority for VCP Register

Offset: 0x8 | Read/Write: R/W | Reset: 0x12201209 (0bxx010010001000xxx001001000001001)

Bit	Reset	Description
29:27	0x2	VDE: 000=Highest Priority 111=Lowest Priority 0 = HIGHEST

Bit	Reset	Description
		1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
26:24	0x2	HOST1X: 000=Highest Priority 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
23:21	0x1	APB: 11=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = LOWEST
20:18	0x0	PSB: 11=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = LOWEST
14:12	0x1	IROM0: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
11:9	0x1	IRAM3: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
8:6	0x0	IRAM2: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
5:3	0x1	IRAM1: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6

Bit	Reset	Description
		7 = LOWEST
2:0	0x1	IRAM0: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST

38.6.4 ARB_PRIO_DMA_PRIORITY_0

Shared Resource Priority for DMA Register

Offset: 0xc | Read/Write: R/W | Reset: 0x1b20365b (0bxx011011001000xxx011011001011011)

Bit	Reset	Description
29:27	0x3	VDE: 000=Highest Priority 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
26:24	0x3	HOST1X: 000=Highest Priority 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
23:21	0x1	APB: 11=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = LOWEST
20:18	0x0	PSB: 11=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = LOWEST
14:12	0x3	IROM0: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
11:9	0x3	IRAM3: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2

Bit	Reset	Description
		3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
8:6	0x1	IRAM2: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
5:3	0x3	IRAM1: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
2:0	0x3	IRAM0: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST

38.6.5 ARB_PRIO_UCQ_PRIORITY_0

Shared Resource Priority for UCQ Register

Offset: 0x14 | Read/Write: R/W | Reset: 0x24204924 (0bxx100100001000xxx100100100100100)

Bit	Reset	Description
29:27	0x4	VDE: 000=Highest Priority 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
26:24	0x4	HOST1X: 000=Highest Priority 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
23:21	0x1	APB: 11=Lowest Priority 0 = HIGHEST

Bit	Reset	Description
		1 = PRIORITY1 2 = PRIORITY2 3 = LOWEST
20:18	0x0	PSB: 11=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = LOWEST
14:12	0x4	IROM0: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
11:9	0x4	IRAM3: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
8:6	0x4	IRAM2: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
5:3	0x4	IRAM1: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
2:0	0x4	IRAM0: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST

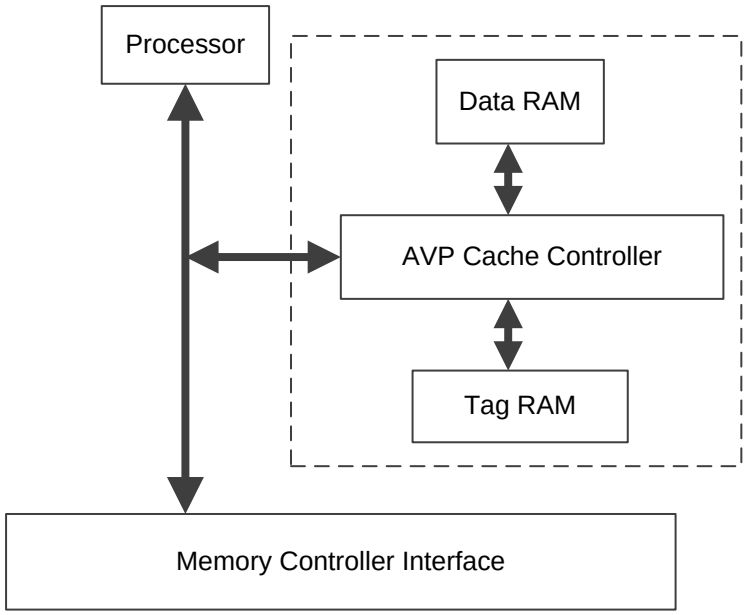
39.0 AVP CACHE CONTROLLER

39.1 Overview

The AVP Cache Controller is used to cache both instructions and data stored in main memory for the ARM7™ processor. It provides cached and uncached accesses to memory for the AVP. Cacheable and uncacheable address ranges are determined by descriptors associated with linear segments of addresses. Memory Controller read and write efficiency is increased through use of read and write buffers, respectively. Refer also to the Audio-Video Processor (AVP) section of this document.

The following diagram shows a high-level view of the AVP Cache Controller.

Figure 135: Cache Controller Overview



39.2 Features

- Four-way set associative cache
- 32 KB cache size
- 32-byte cache line size
- 32 segment descriptors
- Atomic updates for arbitrary sets of descriptors in the MMU

39.3 Address Space

The AVP cache handles addresses in one of the following ranges (statically defined by the global address map):

- A window corresponding to most of the external memory, [0x40:2G[. The exception vectors are located in [0, 0x40[.
- A window corresponding to the GART [GART_BASE, GART_BASE + GART_SIZE[. Refer to the Address Map section for the GART address range.
- A window corresponding to the register interface

All other addresses are routed towards the crossbar and the rest of the system; they are ignored by the AVP cache. Address decode and data return multiplexing are not part of the AVP cache but of a global address decoder.

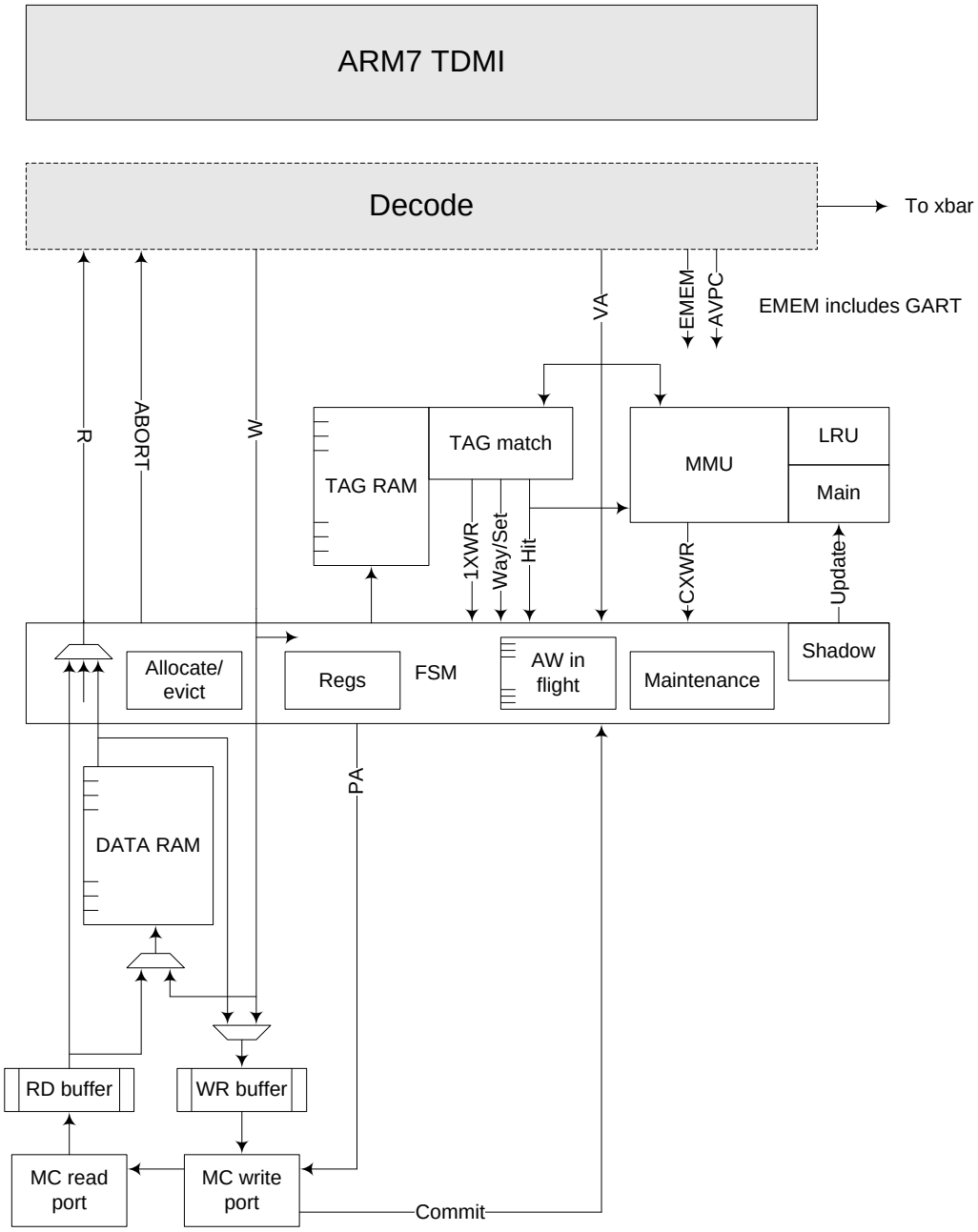
39.4 Reference Block Decomposition

The main blocks inside the AVP cache are:

- Read buffer
- Write buffer
- Tag and Data RAM
- MMU
- Tag check logic
- A main control FSM

The next figure shows a reference decomposition of AVP cache in functional blocks. The grey blocks are not part of the AVP Cache Controller.

Figure 136: AVP Cache Reference Functional Decomposition



39.5 Memory Management Unit

The Memory Management Unit (MMU) associates each memory address in the EMEM address range (MC and GART) with a set of attributes. The set of attributes is defined by a bitmap with the following fields:

- C: Cacheable, indicates that the corresponding address may be stored in the cache
- X: eXecute, indicates that the corresponding address is allowed to be the target of an instruction fetch
- R: indicates that the corresponding address is allowed to be the target of a read data access
- W: indicates that the corresponding address is allowed to be the target of a write data access

An access that is not identified as being allowed results in an abort and the characteristics of the aborted transaction captured. The exact transaction parameters captured are:

- Type of access (X, R, or W)
- Target address
- Sequential or Initial
- Width of access (8, 16, or 32 bits)
- Associated segment descriptor (including an identification for the default descriptor and for multiple segments)

39.5.1 Segment Descriptors

The memory attributes are defined as constant on contiguous segments of the address space. The segments are aligned on cache line boundaries, with each segment defined by:

- The set of attributes associated with the segment: C, X, R, W
- The start address of the segment (the LSBs are not stored)
- The stop address of the segment (the LSBs are not stored)

An address matches a segment descriptor if $Start[31:5] \leq Address[31:5] \leq Stop[31:5]$ so that a descriptor with Stop less than Start never matches any address. This is why there is no explicit valid bit, but the hardware maintains a bitmap of valid descriptors so as to avoid matching invalid descriptors.

39.5.2 Atomic Update of Segment Descriptors

Correct software operation requires the capability to perform atomic updates of multiple segment descriptors. Atomic operations are cumbersome for software, instead a simple hardware context switch approach is implemented. Note that the atomic update described below is the only way to update a segment descriptor; that is, there is no direct way to modify a descriptor stored in the main descriptor table.

The MMU contains a shadow table of segment descriptors and an associated bitmap of update flags, one per table position. When triggered, hardware copies from the shadow table to the main table the set of descriptors identified in the bitmap. No MMU match is performed until the update is complete, backpressuring the ARM7™ bus, if needed. At the end of the copy operation, all update flags must be cleared.

At any update trigger, even if the update bitmap is empty, the Least Recently Used Table Lookaside Buffer (LRU TLB) is invalidated.

39.6 Access to Memory

39.6.1 Read Buffer

The Tegra® K1 AVP cache introduces a read buffer that captures the full memory atom returned by the memory controller. Subsequent linear accesses can hit in the read buffer, increasing both performance and efficiency.

The operation of the read buffer is (almost) transparent for software by using the following buffer flushing conditions:

- The read buffer is flushed for any external memory data read access that is not sequential with the previous external memory data read access in the current scan direction. In particular, this precludes:
  - Using any byte in the buffer more than once (this avoids problems with algorithms that poll on an uncached address)
  - Non-contiguous accesses in the buffer
- Any write that overlaps with the read buffer
- Any locked access
- Any change of any interrupt
- Any change inside the MMU that might impact any memory attribute
- Any cache line eviction
- If a time-to-live timer expires, the timer is started at the time the read buffer is initialized and restarted at each hit in the buffer
- When explicitly triggered by a control bit

39.6.2 Write Merging Buffer

The AVP cache introduces a write buffer that captures up to 256 bits corresponding to a cache line for better performance and efficiency. The operation of the write buffer is transparent for software by using the following buffer flushing conditions:

- The write buffer is flushed for any external memory data write access that is not sequential with the previous external memory data write access in the current scan direction. In particular, this precludes:
  - Using any byte in the buffer more than once (this avoids problems with algorithms that poll on an uncached address, ensuring all changes will be observed in external memory)
  - Non-contiguous accesses in the buffer
- Any read that overlaps with the write buffer
- Only MC atoms with at least one valid byte in them are forwarded to the MC. If more than one MC atom is valid, they are sent coalesced to the MC.
- Any locked access
- Any change of any interrupt
- Any change inside the MMU that might impact any memory attribute
- If a time-to-live timer expires, the timer is started at the time the write buffer is initialized and restarted at each hit in the buffer.
- When explicitly triggered by a control bit

Note: The write buffer is not used when any non-blocking cache operation is active.

39.7 Cache Maintenance

The Cache Maintenance Control (CMC) block provides visibility of the cache operation to software; it allows software-controlled maintenance operations. Caches maintain a lot of state information, so direct manipulation of registers or memory records inside the cache is risky. In general, hardware provides higher level primitives that are safe to use with a minimal software involvement.

The general primitive operation in the CMC works like this:

- A set of registers is programmed detailing the action to be performed
- A specific register access triggers a hardware state machine that performs the action
- Hardware explicitly signals completion of the action
- External access to involved registers before completion of the action might result in undefined behavior

The following maintenance operations are supported in the CMC:

- Clean line: if the line is dirty write the line to main memory, mark the line as clean
- Invalidate line: mark the line as invalid, this is done without checking the dirty bits associated with the line
- Clean and invalidate line: if the line is dirty, write the line to main memory, and mark the line as invalid
- Clean ways, invalidate ways, clean and invalidate ways: same as above but applied on all lines in a set of ways identified by a bitmap. Invalidate ways is useful at reset; clean and clean and invalidate are useful for coherency, especially before a suspend to memory

39.8 Initializing the AVP Cache Controller

To initialize the AVP cache controller:

1) Write INIT to the AVP_CACHE_MMU_CMD_0 register.

2) Write AVP_CACHE_MMU_FALLBACK_ENTRY_0 with the following bit settings:

- CACHED = DISABLE
- EXE_ENA = ENABLE
- RD_ENA = ENABLE
- WR_ENA = ENABLE

3) Initialize the AVP_CACHE_MMU_SHADOW_ENTRY_0 registers using the sequence described in the AVP_CACHE_MMU_SHADOW_ENTRY_0 register description.

4) Invalidate the cache:

- Write 0x1 to AVP_CACHE_INT_CLEAR_0.
- Write AVP_CACHE_MAINT_2_0 with the following bits:
  - OPCODE = INVALID_WAY
  - WAY_BITMAP = DEFAULT_MASK
- Poll AVP_CACHE_INT_RAW_EVENT_0 until bit 0 goes high
- Write AVP_CACHE_INT_CLEAR with the value read from AVP_CACHE_INT_RAW_EVENT_0

5) Enable the cache by writing AVP_CACHE_CONFIG_0 with the following bits:

- ENABLE_CACHE = TRUE
- TAG_CHECK_ABORT_ON_ERROR = 1

39.9 AVP Cache Controller Registers

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

39.9.1 AVP_CACHE_CONFIG_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x80004000 (0b1xxxxxxxxxxxxxx00100000000xx0000) | Default: 0x00000001

Bit	Reset	SW Default	Description
31	0x1	_NONE_	OBS_BUS_EN: Enable clocking of the OBS_BUS capture register. This has a default value of TRUE to allow monitoring things before registers can be programmed. 0 = FALSE 1 = TRUE
16	FALSE	_NONE_	DISABLE_SAMELINE: enable the feature that when consecutive reads are within the same cache line, reading TAG RAM can be avoided to save power. 0 = FALSE 1 = TRUE
15	0x0	_NONE_	TAG_CHECK_CLR_ERROR
14	0x1	_NONE_	TAG_CHECK_ABORT_ON_ERROR
13	0x0	_NONE_	FULL_LINE_DIRTY: TRUE: if a word of a cache line is dirty, assume the full line is dirty and write out 256 bits. Do not write out just 128 bits if all dirty words fall within the same 128 bits. This bit is really only there to check out how much power is really saved by this feature. 0 = FALSE 1 = TRUE
12	0x0	_NONE_	ENABLE_HANG_DETECT: TRUE: when the request state machine doesn't return to idle within a specific number of cycles, the cache generates an ABORT to the ARM7. This is useful for debugging. 0 = FALSE 1 = TRUE
11	0x0	_NONE_	DISABLE_RB: TRUE: all uncached reads are from external memory. The 'cached' 128-bit read buffer is not used. Asserting this bit when the read buffer is active will immediately invalidate it. 0 = FALSE 1 = TRUE
10	0x0	_NONE_	DISABLE_WB: TRUE: don't gather sequential writes in the write buffer. Always write them immediately to external memory. Asserting this bit while the write buffer is active, will immediately flush its contents. 0 = FALSE 1 = TRUE
9:8	PARALLEL	_NONE_	MMU_TAG_MODE: Determines when the cache will check the TAG RAM and the MMU. PARALLEL: when a request comes in, both TAG and MMU lookup's are done right away. This results in the fastest operation. The result that comes up first is used. When both results come in at the same time, the MMU values are used. TAG_FIRST: look up the MMU first. Only look up MMU when that the TAG lookup results in a MISS MMU_FIRST: look up the MMU first. Only look up TAG when the segment is cached. NOTE: in PARALLEL and TAG_FIRST mode, lookups are kicked off for all ARM requests, even if they don't fall inside the GART or the EMEM address space, so they will consume more power than MMU_FIRST. In case the ARM is running most of its code from IRAM (which are often the most power sensitive use cases), it is better to use MMU_FIRST and take a 1 cycle performance penalty for cache hits. 0 = PARALLEL 1 = TAG_FIRST 2 = MMU_FIRST
7	0x0	_NONE_	ENABLE_INTERRUPT: 0 = FALSE

Bit	Reset	SW Default	Description
			1 = TRUE
6	0x0	_NONE_	NEVER_ALLOCATE: TRUE: when there's a cacheable access that results in a cache miss, no new line is allocated. The access is treated as an uncached access instead. The effect is similar to locking all the ways of a cache (see LOCK_BITMAP). 0 = FALSE 1 = TRUE
3	0x0	_NONE_	FORCE_WRITE_THROUGH: FALSE: cached writes are Write Back - Write Allocate. TRUE: cached writes are Write Through. When the memory location is already in the cache, the data will be written both to external memory and to the cache line. The line will not be marked dirty. When the memory location is not in the cache, the write will be treated as an uncached write. 0 = FALSE 1 = TRUE
2	0x0	_NONE_	DISABLE_RANDOM_ALLOC: FALSE: an LSFR is used to randomize the way that will be allocated next. TRUE: the LSFR is strapped to zero. This may be useful for testing purposes to remove an element of randomness. 0 = FALSE 1 = TRUE
1	0x0	_NONE_	ENABLE_SKEW_ASSOC: ENABLE: enable skewed associativity when calculating the indices of the different tag rams. This can avoid some pathological cache collision cases. DISABLE: no skewed associativity. The tag indices are taken straight from the transaction address. Note: DO NOT ENABLE. 0 = FALSE 1 = TRUE
0	0x0	TRUE	ENABLE_CACHE: TRUE: activates the cache. Memory requests inside the GART or EMEM address space are cached depending on the outcome of the MMU. FALSE: cache is inactive. All memory requests that fall inside the GART or EMEM address space are sent directly to external memory. The MMU is not checked at all when the cache is disabled (and does not have to be initialized). Read buffer and write buffer optimizations are performed. 0 = FALSE 1 = TRUE

39.9.2 AVP_CACHE_LOCK_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	LOCK_BITMAP: When bit is set to 1, the corresponding way is locked and never considered for cache line replacement.

39.9.3 AVP_CACHE_SIZE_0

Hardcoded value that indicates the number of ways.

Offset: 0xc | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	MAX_WAY_INDEX: Number of ways – 1. So 0 -> 1 way, 1 -> 2 ways, etc. This field is hardcoded to value 3, indicating a 4-way cache.

39.9.4 AVP_CACHE_LFSR_0

Offset: 0x10 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	STATUS: For cache line selection during cache line fills, the cache uses a LFSR to determine the next way (in case there are no unallocated ways for that particular line.) This field contains the current status of the LFSR pseudo random generator. It gets updated for each cache fill operation. Only useful for debugging.

39.9.5 AVP_CACHE_TAG_STATUS_0

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

Bit	Reset	Description
31:5	X	CONFLICT_ADDR: Physical address for which a TAG_CHECK_ERROR was generated.
0	X	TAG_CHECK_ERROR: When asserted, there was a cache tag lookup with a hit for more than 1 way for the same tag address. This would be an internal error that is never supposed to happen.

39.9.6 AVP_CACHE_CLKEN_OVERRIDE_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	CLK_GATED	RD_MCCIF_CLKEN_OVR: This overrides the clock gating of the memory read block that is used by the cache. This bit should always be set to CLK_GATED to ensure MCCIF clock shutdown when the cache is idle. 0 = CLK_GATED 1 = CLK_ALWAYS_ON
0	CLK_GATED	WR_MCCIF_CLKEN_OVR: This overrides the clock gating of the memory write block that's used by the cache. This bit should always be set to CLK_GATED to ensure MCCIF clock shutdown when the cache is idle. 0 = CLK_GATED 1 = CLK_ALWAYS_ON

39.9.7 AVP_CACHE_MAINT_0_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	ADDR: Address field used for maintenance operations that require an address. Depending on the kind of operation, the semantics of this field may be a physical address in external memory (e.g., CLEAN_PHY and INVALID_PHY) or it may be the address of a line. For other operations, such as CLEAN_WAY, it is not used at all. This register needs to be written before the actual maintenance operation is executed by writing to MAINT_2.

39.9.8 AVP_CACHE_MAINT_1_0

Offset: 0x24 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA: Data field used for maintenance operations that require a data parameter.

39.9.9 AVP_CACHE_MAINT_2_0

All maintenance requests are blocking operations: new request from the ARM7 will stall until the maintenance operation has completed, so the ARM does not need to monitor the MAINTENANCE_DONE bit to ensure that everything has completed. For all *_WAY operations, this can result in seriously long stalls. In cases where latency is important (e.g., for interrupts), *_WAY

maintenance operations should be avoided. *_PHY or *_LINE maintenance operations should be used instead. The decision to use *_PHY or *_LINE will depend on the size physical memory buffer that needs to be cleaned or invalidated from the cache. For very large physical buffers or when the full cache needs to be cleaned or invalidated, software should simply loop over all lines in all ways and run the *_LINE command on each of them. When a small physical buffer needs to be cleaned and/or invalidated from the cache, it is more effective to use *_PHY commands.

Offset: 0x28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:8	0x0	WAY_BITMAP: Select which the ways on which the *_WAY maintenance operations will work. One bit for each way.
7:0	NOP	OPCODE: 0 = NOP : No operation. after any op is executed, the field should be reset to NOP. 1 = CLEAN_PHY : Clean by physical address. This operation checks that the physical address stored in MAINT_0.ADDRESS is currently stored in the cache and will write its modified contents back to external memory. The cache line is not invalidated. 2 = INVALID_PHY : Invalidate by physical address. If the cache line was dirty, the contents are NOT first written to external memory. 3 = CLEAN_INVALID_PHY : Clean AND invalidate by physical address. CLEAN_PHY followed by INVALID_PHY. 9 = CLEAN_LINE : Clean 1 line in the cache. The line to be cleaned is taken from the MAINT_0.ADDRESS field and of the format { WAY_NR, LINE_NR }. 10 = INVALID_LINE : Invalidate 1 line in the cache. 11 = CLEAN_INVALID_LINE : Clean and invalidate 1 line in the cache. 17 = CLEAN_WAY : Clean all the lines of all the ways that are selected in the WAY_BITMAP field. 18 = INVALID_WAY : Invalidate all the lines of all the ways that are selected in the WAY_BITMAP field. 19 = CLEAN_INVALID_WAY : Clean AND invalidate all the lines of all the ways that are selected in the WAY_BITMAP field.

39.9.10 AVP_CACHE_INT_MASK_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	MAINTENANCE_ERROR: Enable interrupt for MAINTENANCE_ERROR
0	0x0	MAINTENANCE_DONE: Enable interrupt for MAINTENANCE_DONE

39.9.11 AVP_CACHE_INT_CLEAR_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	MAINTENANCE_ERROR: Clear both RAW_EVENT and INT_STATUS
0	0x0	MAINTENANCE_DONE: Clear both RAW_EVENT and INT_STATUS

39.9.12 AVP_CACHE_INT_RAW_EVENT_0

Offset: 0x48 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	MAINTENANCE_ERROR: Raw event for MAINTENANCE_ERROR. Raised when a MAINTENANCE operation was no successful. Usually because an incorrect OPCODE was written to the MAINT_2 register. When this bit is asserted, MAINTENANCE_DONE is not asserted.
0	X	MAINTENANCE_DONE: Raw event for MAINTENANCE_DONE. Raised when a MAINTENANCE operation has successfully completed.

39.9.13 AVP_CACHE_INT_STATUS_0

Offset: 0x4c | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	MAINTENANCE_ERROR: Interrupt status (masked) for MAINTENANCE_ERROR
0	X	MAINTENANCE_DONE: Interrupt status (masked) for MAINTENANCE_DONE

39.9.14 AVP_CACHE_RB_CFG_0

Offset: 0x80 | Read/Write: R/W | Reset: 0x00000100 (0bxxxxxxxxxxxxxxxxxxxxxxx0100000000)

Bit	Reset	Description
9:0	0x100	TTL_TIMER_MAX: Time to live counter for uncached memory reads in the read buffer. This timer starts as soon as data from an external memory read has arrived in the read buffer. When this counter expires, the data in the read buffer is marked in valid. A value of 0 disables the TTL timer. It does NOT disable the read buffer! Use CONFIG.DISABLE_RB to disable the read buffering instead.

39.9.15 AVP_CACHE_WB_CFG_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00000100 (0bxxxxxxxxxxxxxxxxxxxxxxx0100000000)

Bit	Reset	Description
9:0	0x100	TTL_TIMER_MAX: Time to live counter for uncached memory writes in the write coalescing buffer. This time starts as soon as the first uncached write data is received from the ARM. When the timer expires, the contents of the write buffer are written to external memory. (There are many other events that may result in the write buffer being flushed to memory, such as non-sequential writes, RAW hazards, etc.) A value of 0 disables the TTL timer. It does NOT disable the write buffer. Use CONFIG.DISABLE_WB to disable write buffering instead.

39.9.16 AVP_CACHE_MMU_FALLBACK_ENTRY_0

Offset: 0xa0 | Read/Write: R/W | Reset: 0x0000000e (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1110)

Bit	Reset	Description
3	ENABLE	WR_ENA: 0 = DISABLE 1 = ENABLE
2	ENABLE	RD_ENA: 0 = DISABLE 1 = ENABLE
1	ENABLE	EXE_ENA: 0 = DISABLE 1 = ENABLE
0	DISABLE	CACHED: 0 = DISABLE 1 = ENABLE

39.9.17 AVP_CACHE_MMU_SHADOW_COPY_MASK_0_0

Offset: 0xa4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	MASK

39.9.18 AVP_CACHE_MMU_CFG_0

Offset: 0xac | Read/Write: R/W | Reset: 0x000000X7 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0111)

Bit	Reset	Description
5	X	CLR_ABORT: 0: NOP 1: Clean the abort statistics. (See ABORT_MODE-STORE_FIRST)
4	X	ABORT_MODE: STORE_FIRST: store the stats of the first MMU abort. Later aborts do not overwrite the first one as long as the abort statistics have not been cleared. STORE_LAST: always store the statistics of the latest abort. 0 = STORE_FIRST 1 = STORE_LAST
3	DISABLE	SEG_CHECK_ALL_ENTRIES: DISABLE: stop doing the lookup as soon as an entry is found that matches. This is the fastest and preferred way, but it will not detect a MMU entry overlap error if those overlapping entries are not part of the same group of 4. ENABLE: all lookups will go through all entries in the MMU. This will impact performance, but is guaranteed to flag overlapping address ranges between MMU entries. 0 = DISABLE 1 = ENABLE
2	ENABLE	TLB_ENA: Enable MMU TLB lookup optimization 0 = DISABLE 1 = ENABLE
1	ENABLE	SEQ_ENA: Enable MMU sequential access optimization 0 = DISABLE 1 = ENABLE
0	ENABLE	BLOCK_MAIN_ENTRY_WR: Enable writing directly into the active MMU entries instead of going through the shadow entries first and doing a block copy. 0 = DISABLE 1 = ENABLE

39.9.19 AVP_CACHE_MMU_CMD_0

Offset: 0xb0 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	CMD: 0 = NOP 1 = INIT 2 = COPY_SHADOW

39.9.20 AVP_CACHE_MMU_ABORT_STAT_0

Offset: 0xb4 | Read/Write: RO | Reset: 0x00XX0XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
21	X	PROT: 0 = FALSE 1 = TRUE
20	X	SEQ: 0 = FALSE 1 = TRUE
19:18	X	SIZE
17:16	X	TYPE: 0 = EXE 1 = RD 2 = WR

Bit	Reset	Description
8:4	X	ENTRY
3	X	OVERLAP: 0 = FALSE 1 = TRUE
2:0	X	UNIT: 0 = NONE 1 = CACHE 2 = SEQ 3 = TLB 4 = SEG 5 = FALLBACK

39.9.21 AVP_CACHE_MMU_ABORT_ADDR_0

Offset: 0xb8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ADDR

39.9.22 AVP_CACHE_MMU_ACTIVE_ENTRIES_0_0

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

Bit	Reset	Description
31:0	X	ENTRIES

39.9.23 AVP_CACHE_MMU_SHADOW_ENTRY_0

How to program the MMU entries:

The full MMU table should be updated atomically. The procedure to do this is as follows:

- Write all new entries in the MMU_SHADOW_ENTRY table
- For each of the entries in the shadow table that need to be copied, set the corresponding bit in the MMU_SHADOW_COPY_MASK register
- Issue the COPY_SHADOW command via the MMU_CMD register.

MMU_SHADOW_ENTRY addresses are mapped as follows: for each MMU entry, there are 4 32-bit words reserved, so to get the base pointer of an entry, you need to take the address of MMU_SHADOW_ENTRY_0 + (entry_nr * 16).

From there, the following offsets apply:

- 32-bit word 0: MIN_ADDR
- 32-bit word 1: MAX_ADDR
- 32-bit word 2: CXRW attributes

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

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

Bit	Reset	Description
Word 0		
31:5	X	MIN_ADDR
Word 1		
31:5	X	MAX_ADDR
Word 2		
3	X	WR_ENA: 0 = DISABLE 1 = ENABLE
2	X	RD_ENA: 0 = DISABLE 1 = ENABLE
1	X	EXE_ENA: 0 = DISABLE 1 = ENABLE
0	X	CACHED: 0 = DISABLE 1 = ENABLE

39.9.24 AVP_CACHE_MMU_MAIN_ENTRY_0

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

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

Bit	Reset	Description
Word 0		
31:5	X	MIN_ADDR
Word 1		
31:5	X	MAX_ADDR
Word 2		
3	X	WR_ENA: 0 = DISABLE 1 = ENABLE
2	X	RD_ENA: 0 = DISABLE 1 = ENABLE
1	X	EXE_ENA: 0 = DISABLE 1 = ENABLE
0	X	CACHED: 0 = DISABLE 1 = ENABLE

[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